

JOURNAL OF APPLIED  
COMPUTER SCIENCE  
Vol. xx No. xx (xxxx), pp. xx-xx

# Object-Oriented Internet - Azure Interoperability

Mariusz Postół<sup>1</sup>, Piotr Szymczak<sup>1</sup>, Clemens Vasters<sup>2</sup>

<sup>1</sup>*Lodz University of Technology  
Institute of Information Technology  
ul. Wólczańska 215, 90-924 Lodz, Poland  
mailto:mariusz.postol@p.lodz.pl*

<sup>2</sup>*Microsoft  
Faculty/Department/Office Name  
Postal Address with the zip-code  
clemensv@microsoft.com*

**Abstract.** *Each paper should be followed by a compact abstract which points to the main scopes and the results obtained by the paper. The abstract should be written with the Times New Roman font, 10 pt, justified, and with the 1cm margin both left and right side with respect to the margin of the paper. The abstract should not contain any formulas or references, and should not exceed 200 words.*

**Keywords:** *Azure, Cloud Computing, Object-Oriented Internet, Industrial communication, Industry 4.0, Internet of Things, Machine to Machine Communication, OPC Unified Architecture*

## 1. Introduction

All the time, the Information and Communication Technology is providing society with a vast variety of new distributed applications aimed at micro and macro

optimization of the industrial processes. Obviously, the design foundation of this kind of application has to focus primarily on communication technologies. Based on the role humans take while using these applications they can be grouped as follows:

- **human-centric** - information origin or ultimate information destination is an operator,
- **machine-centric** - information creation, consumption, networking, and processing are achieved entirely without human interaction.

A typical **human-centric** approach is a web-service supporting, for example, a web user interface (UI) to monitor conditions, and manage millions of devices and their data in a typical cloud-based IoT approach. It is characteristic that, in this case, any uncertainty and necessity to make a decision can be relaxed by human interaction. Coordination of robots behavior in a work-cell (automation islands) is a **machine-centric** example. In this case, it is essential that any human interaction is impractical or even impossible. This interconnection scenario requires the machine to machine communication (M2M) demanding multi-vendor devices integration.

From the M2M communication concept, a broader concept of a smart factory can be derived. In this concept, the mentioned robots are only executive assets of an integrated supervisory control system responsible for macro optimization of an industrial process composed into one whole. Deployment of the smart factory concept requires a hybrid solution and interoperability of the mentioned above heterogeneous environments. This approach is called the fourth industrial revolution and coined as Industry 4.0. It is worth stressing that machines - or more general assets - interconnection is not enough, and additionally, assets interoperability has to be expected for the deployment of this concept. In this case, multi-vendor integration makes communication standardization especially important, namely it is required that the payload of the message is standardized to be factored on the gathering site and consumed on the ultimate destination site.

Highly-distributed solutions used to control real-time process aggregating islands of automation (e.g. virtual power plants producing renewable energy) additionally must leverage public communication infrastructure, namely the Internet. Internet is a demanding environment for highly distributed process control applications designed atop the M2M communication paradigm because

- it is a globally shareable environment and can be also used by malicious

users

- it offers only non-deterministic communication making integration of islands of automation designed against the real-time requirements a difficult task

Today both obstacles can be overcome, and as examples, we have bank account remote control and voice over IP in daily use. The first application must be fine-tuned in the context of data security, and the second is very sensitive on time relationships. Similar approaches could be applied to adopt the well known in process control industry concepts:

- Human Machine Interface (HMI)
- Supervisory Control and Data Acquisition (SCADA)
- Distributed Control Systems (DCS)

A detailed examination of these solutions is far beyond the scope of this article. It is only worth stressing that, by design, all of them are designed on the foundation of interactive communication. Interactive communication is based on a data polling foundation. In this case, the application must follow the interactive behavioral model, because it actively polls the data source for more information by pulling data from a sequence that represents the process state in time. The application is active in the data retrieval process - it controls the pace of the retrieval by sending the requests at its convenience. Such a polling pattern is similar to visiting the books shop and checking out a book. After you are done with the book, you pay another visit to check out another one. If the book is not available you must wait, but you may read what you selected. The client/server archetype is well suited for the mentioned above applications.

After dynamically attaching a new island of automation the control application (responsible for the data pulling) must be reconfigured for this interoperability scenario. In other words, in this case, the interactive relationship cannot be directly applied because the control application must be informed on how to pull data from a new source. As a result, a plug and produce scenario cannot be seamlessly applied. A similar drawback must be overcome if for security reasons suitable protection methods have been applied to make network traffic propagation asymmetric. It is accomplished using intermediary devices, for example, firewalls, to enforce traffic selective availability based on predetermined security rules against unauthorized access.

Going further, we shall assume that islands of automation are mobile, e.g. autonomous cars passing a supervisory controlled service area. In this case, the behavior of the interconnected assets is particularly important concerning the environment in which they must interact. This way we have entered the Internet of Things domain of Internet-based applications.

If we must bother with the network traffic propagation asymmetry or mobility of the asset network attachment-points the reactive relationship could relax the problems encountered while the interactive approach is applied. In this case, the sessionless publisher-subscriber communication archetype is a typical pattern to implement the abstract reactive interoperability paradigm. The sessionless archetype is a message distribution scenario where senders of messages, called publishers, do not send them directly to specific receivers, called subscribers, but instead, categorize the published messages into topics without knowledge about which subscribers if any, there may be. Similarly, subscribers express interest in one or more topics and only receive messages that are of interest, without knowledge about which publishers, if any, there are. In this scenario, the publishers and subscribers are loosely coupled, i.e. they are decoupled in time, space and synchronization [1].

If the **machine-centric** interoperability - making up islands of automation - must be monitored and/or controlled by a supervisory system cloud computing concept may be recognized as a beneficial solution to replace or expand the mentioned above applications, i.e. HMI, SCADA, DCS, etc. Cloud computing is a method to provide a requested functionality as a set of services. There are many examples that cloud computing is useful to reduce costs and increase robustness. It is also valuable in case the process data must be exposed to many stakeholders. Following this idea and offering control systems as a service, there is required a mechanism created on the service concept and supporting abstraction and virtualization - two main pillars of the cloud computing paradigm. In the cloud computing concept, virtualization is recognized as the possibility to share the services by many users, and abstraction hides implementation details.

Deployment of the hybrid solution providing interoperability of the **machine-centric** cyber-physical systems and **human-centric** cloud-based front-end can be implemented applying the following scenarios:

- **direct interconnection** - cloud-based dedicated communication services allow to attach it to the cyber-physical system making up a consistent M2M communication network using an in-bound protocol stack

- **gateway based interconnection** - typical build-in communication services allows to attach the cloud computing to the cyber-physical system using an out-of-bound protocol stack

By design, the direct approach requires that the cloud has to be compliant with the interoperability standard the cyber-physical system is built around - it becomes a consistent part of the cyber-physical system. Data models, roles, and responsibility differences of both solutions make this approach impractical and impossible to be applied in typical cases. A more detailed description is covered by the section **Azure to Sensors (A2S) connectivity deployment**.

This article addresses further research on the integration of the cyber-physical systems in the context of new emerging disciplines, i.e. Industry 4.0 (I4.0) and the Internet of Things (IoT). The new architecture is proposed for integration of the multi-vendor **machine-centric** cyber-physical system designed atop of M2M reactive communication and emerging cloud computing as a **human-centric** front-end. To support the multi-vendor environment OPC Unified Architecture interoperability standard has been selected. The proposals are backed by proof of concept reference implementations. Prototyping addresses Microsoft Azure Cloud as an example. The prototyping outcome has been just published on GitHub as the open-source (MIT licensed). The proposed solutions have been harmonized with the more general concept called the Object-Oriented Internet.

The main goal of this article is to provide proof that:

1. reactive interoperability M2M communication based on the OPC UA standard can be implemented as a powerful standalone library without dependency on the Client/Server session-oriented archetype
2. Azure interoperability can be implemented as an external part employing out-of-band communication without dependency on the OPC UA implementation
3. the proposed generic architecture allows that the gateway functionality is composable at run-time - no programming required

The remainder of this paper is structured as follows. Sect. 2 analyzes data presentation user interface, available native communication services, and data/metadata model offered by the Microsoft Azure. The discussion covered by this section is the foundation for selecting services utilized to expose process data and suitable protocol stack to support interconnection. In Sect. 3 the discussion focuses on the generic architecture that is to be used as a foundation for further decisions

addressing the systematic design of the interoperability of the cyber-physical systems and cloud-based front-end. Sect. 4 presents the proposed open and reusable software model. It promotes a reactive interoperability pattern and a generic approach to establishing interoperability-context. A reference implementation of this archetype is described in Sect. 5. The most important findings and future work are summarized in Sect. 6.

## 1.1. Consider to add

### 1.1.1. Scope

What we must do to prove the goal have been achieved. Extent or range of development, view, outlook, application, operation, effectiveness, etc.

### 1.1.2. Related work

Any information about available reusable deliverables related to this work.

## 2. Azure Main Technology Features

### 2.1. Services

Deployment of the hybrid solution providing interoperability of the **machine-centric** cyber-physical systems designed atop of M2M reactive communication and emerging cloud computing as a **human-centric** front-end requires decisions addressing the selection of the services supporting web user interface capable to expose real-time process data. In this context, the service is any autonomous (with own identity) software component or module that is interfacing with selected cyber-physical systems for data collection, analysis, and also remote control. Microsoft Azure is a cloud-based product. It offers a vast variety of services. This virtual environment handles an unlimited number of users and devices organized using a solution concept. The solution aggregates users, devices, services, and required additional resources scoping on a selected scenario. The solution serves as a context that provides a scope to the identifiers (the names of devices, users, process data entities, etc) inside it. Solutions are used to organize deployment entities into logical groups and prevent identity collisions.

The IoT Central service provides a process data visualization user interface. To make this interface meaningful metadata called device template is used to describe devices.

Following the assumption that interconnection between the cyber-physical system and cloud services is designed based on the gateway concept, a middleware must be considered as a coupler. It must be interconnected with the cyber-physical system using an in-band protocol adhering to communications requirements (i.e. protocol profile, data encoding, time relationships, etc.) governing communication of the parts making it up. At the same time, it must support back-and-forth data transfer to the cloud using out-of-band native for the cloud services. The transfer process requires data conversion from source to destination encoding. The IoT Hub is a service hosted in the cloud that supports IoT Central providing a robust messaging solution - it acts as a central message hub for bi-directional communication. This communication is transparent, i.e. it is not data types aware allowing any devices to exchange any kind of data. This service is responsible to manage the devices' identity and it offers the following protocol stacks: AMQP, MQTT, HTTPS.

Before process data can be exposed using a web user interface the data source must be associated with an appropriate solution and validated to make sure that the security rules are not violated. It is hard to assume that the security rules governing the cyber-physical system may also apply to the cloud-based services. In the gateway scenario, they can be mapped on each other or entirely independent. The IoT Hub Device Provisioning Service (DPS) is a helper service for IoT Hub that enables devices' connection process management, upon device providing valid identity attestation it assigns the device to an appropriate IoT Hub instance and returns to the device connection parameters, which allow direct connection with given IoT Hub service. The device proceeds to use the same attestation in IoT Hub connection and based on it, is granted authorization to selected resources and operations including but not limited to data transfer updating the user interface.

It is worth stressing that interaction of the offered by the Azure services can be configured flexibly, and as a result, the presented above selection of services must be recognized as an example only. The IoT Central can be also seamlessly integrated with other services as needed. The following services could also be considered to build cloud-based automation solution:

- **Industrial IoT** - discovering OPC UA enabled servers in a factory network and register them in Azure IoT Hub implemented using IoT Edge

- **Digital Twins** - managing the graph of digital twins, which are to represent some real-world process or entity

**Industrial IoT** promotes OPC UA client/server archetype used to achieve direct and interactive interoperability implemented using IoT Edge services that allow extracting initial data processing to local premises based on the edge concept. **Digital Twins** is an emerging concept to use an observer to replicate selected process state and behavior. The possibility to add value as a result of using these services must be subject to further research.

## 2.2. Data Interchange

System components interoperability means the necessity of the information exchange between them. The main challenge of interoperability implementation is that information is abstract – it is knowledge describing the process in concern state and behavior, e.g. temperature in a boiler, a car speed, an account balance, etc. Obviously, abstraction cannot be processed by the cyber-physical machines. It is also impossible to transfer abstraction from one place to another over the network.

Fortunately, computer science offers a workaround to address that impossibility - the information must be represented as a binary stream. In consequence, we can usually use both ones as interchangeable terms while talking about ICT systems. Unfortunately, these terms must be precisely observed in the context of further discussion, because we must be aware of the fact that the same information could have many different but equivalent representations. In other words, the same information can be represented by a vast variety of different binary patterns. For example, numbers may be represented using 2's Complement and Floating-Point binary representations.

It should be nothing new for us, as it is obvious that the same information printed as a text in regional newspapers in English, German, Polish, etc. does not resemble one another, but the text meaning should always be the same. To understand a newspaper we must learn the appropriate language. To understand the binary data we must have defined a data type – a description of how to create an appropriate bits pattern (syntax) and rules needed to assign the information (semantics), i.e. make any correct bitstream meaningful. Concluding, to make two systems interoperable, a semantic-context must be established. The type plays the role of metadata, a set of data that describes other data. Metadata term is frequently used if the semantic-context is defined using a native language to select built-in types engaging a general-purpose graphical user interface.



Using the data type definitions to describe information interchanged between communicating parties allows:

- Development against a type definition of the user interface
- Implementation of the functionality of the bitstreams conversion in advance

Having defined types in advance, a gateway may provide dedicated conversions functionality, i.e. replacing bitstream used by the cyber-physical system by equivalent one for the cloud-based services. The Azure offers a vast variety of built-in types ready to be used in common cases, but not necessarily there are equivalent counterparts in use by the cyber-physical system. Additionally, the data conversion must address the following issues:

- usually to covert data from source to destination representation, the middle-ware software native types must be used
- if the out of the box set of types is not capable of fulfilling more demanding needs, custom data types must be defined

Although the data conversion is a run-time gateway task the implementation of the conversion algorithms must be recognized as an engineering task, and therefore this topic is not considered for further discussion.

In *IoT Central* a cyber-physical system is represented as a set of devices. The characteristics and behaviors of each device kind are described by the device template. This Device Template (DT) contains also metadata describing the data (called telemetry) exchanged over the wire with the cyber-physical system called Device Capability Model (DCM). Additionally, the DT contains properties, customization, and views definitions used by the service locally. As an option, DCM expressed as a JSON-LD can be imported into a Device Template. *IoT Central* allows also to create and edit a DCM using the dedicated web UI. A JSON file containing DCM can be derived from an information model used as a foundation to establish the semantic-context applied to achieve interoperability of the devices interconnected as the cyber-physical system. DCM development against any external information model is a design-time task and should be supported by dedicated development tools. In any case, the data interchanged between the cloud and the gateway must be compliant with the DCM, hence the middleware software must be aware of conversions that must be applied to achieve this interoperability.

## 2.3. Connectivity

From the cloud side, it is proposed to employ the IoT Hub service to handle the network traffic targeting the cyber-physical system. This service offers profiles of the AMQP, MQTT, HTTPS protocol stacks. In any case, process data (telemetry) is transparently transferred back-and-forth to the upper layer IoT Central service. Hence, the payload formatting is determined by the DCM associated with the IoT Central solution. All the mentioned protocols are standard ones. Consequently, it is possible to apply any available implementation compliant with an appropriate specification to achieve connectivity. In this case, all parameters required to establish connectivity and security-context is up to the external software responsibility. Alternatively, the API offered by the dedicated libraries may be used. Using this API the configuration process can be reduced significantly. Using these libraries, the selection of the communication protocol has an indirect impact on the interoperability features, including performance. The connectivity with IoT Hub, for example, can be obtained using:

- `Microsoft.Azure.Devices` - Service SDK for Azure IoT Devices
- `Microsoft.Azure.Devices.Client` - Device SDK for Azure IoT Hub
- `Microsoft.Azure.Devices.Shared` - Common code for Azure IoT Device and Service SDKs
- `Microsoft.Azure.Devices.Provisioning.Client` - Provisioning Device Client SDK for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Amqp` - Provisioning Device Client AMQP Transport for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Http` - Provisioning Device Client Http Transport for Azure IoT Devices
- `Microsoft.Azure.Devices.Provisioning.Transport.Mqtt` - Provisioning Device Client MQTT Transport for Azure IoT Devices

## 3. OOI Main Technology Features

- Machine To Machine communication based on the semantic data
- OOI PubSub Implementation Architecture
- Simple, complex and structural data processing

## **4. Azure to Sensors (A2S) connectivity deployment (field level connectivity)**

The title must be revised

- **Architecture** - Domain model presenting relationship between the: Azure, PubSub Gateway, Device, Design and development tools
- **Connectivity** - Describe reactive nature of the Azure monitoring process data (telemetry) services.
- **Deployment phases**
  - Design
  - Gateway and devices registration
  - Authentication
  - Device/Service association
  - Device/Application association
  - Establishing session
    - \* Device/Device Template (Device Capability Model) association - establishing a semantic-context
    - \* Security management - establishing security-context
  - Interconnection - exchange of data
  - Maintenance

We have selected IoT Central because:

- provides process data visualization user interface
- allows to describe devices using metadata containing telemetry data types

## **5. Gateway implementation**

- Architecture
- Protocol selection and mapping
- Configuration
- Testing

## 6. Conclusion

The OPC UA PubSub to Azure gateway (AzureGateway) implementation has been just published on GitHub as the open-source (MIT licensed) as a part of the more general concept of the Object-Oriented Internet reactive networking. It is proof of the concept that

1. OPC UA PubSub can be implemented as a powerful standalone package - no C/S dependency
2. Azure interoperability can be implemented as an out-of-band communication (MQTT, AMQP, HTTP) - no PubSub dependency
3. Process data functionality can be composable at run-time - no programming required

## References

- [1] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A. M., *The Many Faces of Publish/Subscribe*, ACM Computing Surveys, Vol. 35, No. 2, 2003, pp. 114–131.