Project information

Project title	Coordinating Optimisation of Complex Industrial Processes
Project acronym	COCOP
Project call	H2020-SPIRE-2016
Grant number	723661
Project duration	1.10.2016-31.3.2020 (42 months)

Document information

Deliverable number	D3.5
Deliverable title	Interface and protocol definitions
Version	1.0
Dissemination level	Public
Work package	WP3
Authors	TUT
Contributing partners	Internal reviewers: MSI, VTT
Delivery date	27.3.2018
Planned delivery month	18
Keywords	Interface, message format, communication protocol

This project has received European Union's Horizon 2020 research and innovation funding under grant agreement No 723661.

VERSION HISTORY

Version	Description	Organisation	Date
0.1.0	Initial draft.	TUT	22.1.2018
0.1.1	Figure captions added.	TUT	23.1.2018
0.1.2	Contents changed in protocols and message formats.	TUT	24.1.2018
0.1.3	Summary chapter modified. Message formats partially rewritten.	TUT	25.1.2018
0.1.4	Added the empty section "Messaging Examples". Added more text about OPC UA.	TUT	15.2.2018
0.1.5	Added messaging examples	TUT	20.2.2018
0.1.6	Minor improvements	TUT	22.2.2018
0.1.7	Typos fixed	TUT	23.2.2018
0.1.8	Suitable Communication Protocols: message patterns explanation improved, a few other modifications in the text Suitable Message Formats: several modifications in the text	TUT	26.2.2018
0.1.9	Minor additions in various places	TUT	28.2.2018
0.1.10	Changes made after internal reviews	TUT	21.3.2018
1.0.0	Minor changes; ready to be delivered	TUT	27.3.2018

EXECUTIVE SUMMARY

This is deliverable D3.5 of the COCOP project (Coordinating Optimisation of Complex Industrial Processes). While a monolithic optimisation task may be performed in a single computational module, decomposed and coordinating optimisation sets extensive requirements on communication. Without communication, the coordination of multiple units is not possible. Furthermore, to enable communication between systems or modules, their mutual integration is an essential requirement.

To enable systems integration, this document covers two core aspects: communication protocols and message formats. Existing legacy systems may provide data access interfaces in various formats, so additional interfaces are specified to enable common formats for interoperability. Interface wrappers or adapters are used for legacy system integration.

This document does not aim at being an exhaustive description, as the design of the exact definitions is still in progress. With this document, the other definition and specifications in WP4 and WP5 can be carried out; and after their progress this definition can be refined. The enhanced definitions will later be published in deliverable D3.7. This document builds on the general architectural principles outlined in D3.1 Software Architecture Description for the Runtime System.

ABBREVIATIONS AND TERMS

AMQP	Advanced Message Queueing Protocol
B2MML	Business to Manufacturing Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OPC UA	Open Platform Communications Unified Architecture
XML	Extensible Markup Language

Communication protocol	A technology to deliver data between logical computing units
Interface	What is required to integrate an information system with another. To have such an interface, two aspects must be covered: a communication protocol and a message format.
Loose coupling	A systems integration approach that aims a minimising the number of direct dependencies between those systems.
Message bus	A medium to deliver messages between network nodes. A message bus typically utilises lower-level communication protocols, building a logical layer of more abstract message delivery. However, in this document, even message buses are considered communication protocols.
Protocol	See "communication protocol"
Serialisation	The operation of generating a textual presentation of a logical object

TABLE OF CONTENTS

1 Suitable Communication Protocols

1.1 Foundations for Choosing Protocols

A message bus is favoured for the message delivery medium to enhance the loose coupling of system components that are inevitably heterogeneous. The protocol of the message bus should enable messages with arbitrary payloads and arbitrary formats (see the following figure).

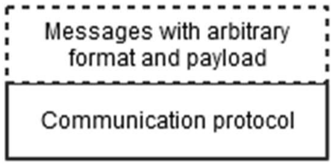

Figure 1 General protocol stack

At least two messaging patterns may be required. First, the message bus should support the publish-subscribe messaging pattern to enable reactive or event-based communication. The pattern has its place where decoupling between network nodes is desirable. For instance, data receivers could sign up for a topic without knowing who actually provides the data. Respectively, the data providers do not know the receivers or how many they are. The approach is well-scalable in terms of a growing network load, because the messaging medium may be scaled according to the needs. Still, regardless of the number of data receivers, a data provider would send each message only once. Second, for some data retrieval scenarios, the request-response pattern may be a more straightforward option. Even in this pattern, the messaging platform may isolate data requestors from data providers. Unfortunately, scalability is reduced, as each data provider must serve each data requestor individually. Fortunately, both messaging patterns may co-exist. For example, to enable request-response on a publish-subscribe medium, the medium may be wrapped with a request-response communication module (see the following figure).

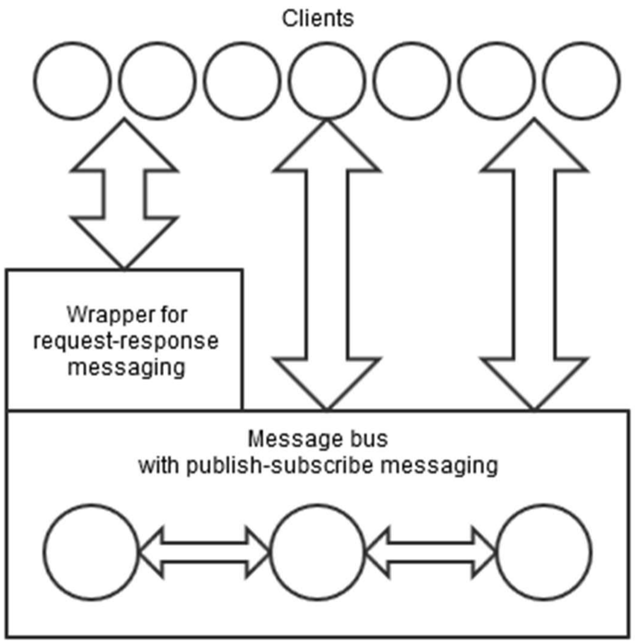
Clients
Wrapper for request-response messaging
Message bus with publish-subscribe messaging

Figure 2 Messaging illustrated

1.2 Protocol and Technology Candidates

The following table summarises the technologies that are considered candidates to implement the required communication platform. However, none of the technologies is likely sufficient alone, but a combination is expected. The following paragraphs elaborate each technology in more detail.

Table 1. Technology candidates for the communication platform implementation

Name	Application area
AMQP (Advanced Message Queueing Protocol) (Aiyagari et al., 2008)	Message bus (plain delivery; centralised)
Apache Kafka (Apache Kafka, 2018)	"Message bus" (stream processor; data delivery and storage)
ZeroMQ (ZeroMQ, 2018)	Message bus (plain delivery; decentralised)
HTTP (Hypertext Transfer Protocol) (Fielding et al., 1999)	Request-response messaging
OPC UA (Open Platform Communications Unified Architecture) (OPC unified architecture specification part 1, 2015)	Integration of the data from production equipment

The message bus candidates include AMQP, Apache Kafka and ZeroMQ. In AMQP, messaging is based on queues. Depending on the message exchange pattern, a queue may be shared between clients, or each client may have its dedicated queues. AMQP has not been designed to store any of the messages; that is, whenever a message is read, it is removed from the queue, and any persistence must be implemented in clients. In contrast, Apache Kafka may inherently store high-volume data. The third of the given candidates, ZeroMQ, has a decentralised approach. It is more lightweight than a centralised bus, but it comes with the cost of fewer coordination features.

HTTP is a good candidate for request-response messaging. Due to its wide adoption in Internet communications, various software tools and libraries support it. Like any communication protocol, a HTTP-based interface may wrap the inherent complexity behind it, such as the publish-subscribe logic utilised in a message bus.

As equipment-related data exchange is related, OPC UA is a good candidate. Various industrial partners have dedicated years of work to specify a family of open specifications to facilitate the integration of production equipment. In this field of application, OPC UA is strong. However, OPC UA is likely not at its best in the delivery of custom-formatted messages, as it also brings additional overhead without facilitating message customisation.

Despite its wide application even in modern industrial plants, OPC DA is not explicitly considered an integration technology candidate. This is due to its dependencies to legacy technology. OPC DA has certain limitations that have led to the development of OPC UA. Thus, to integrate an OPC DA data source, it is preferred to use an appropriate wrapper, e.g., an OPC UA server that reads its data from the OPC DA interface.

Some industrial user interfaces and other systems have a built-in OPC UA client. In such cases, the integration of OPC UA data sources is straightforward, as merely data mapping is required. Thus, it may be appropriate to even wrap the message bus with an OPC UA server. Without such server, the integration approach would be to create an appropriate adapter, which requires more work compared to plain OPC UA data mapping. Similarly, many industrial devices provide OPC UA servers which might be beneficial to connect to the message bus through an adapter.

Although OPC UA traditionally provides a client-server communication model, its "PubSub" candidate specification enables the use of a message bus (OPC unified architecture specification part 14, 2017). Thus, the scalability and distribution advantages of a message bus may be applied even when OPC UA is utilised. However, compared to a generic message bus as the platform, the requirements of OPC UA communication would reduce the freedom of design related to messaging and message contents.

2 Suitable Message Formats

2.1 Message Formats Overview

Suitable message formats include both existing specifications and custom-made formats. Custom-made message formats may be required, because the existing specifications do not probably cover all the requirements of COCOP APIs. However, the exact formats may be specified only after the interface-related requirements are known in more detail. Still, at least in some cases, even existing specifications may be reused.

The message formats should cover at least the needs of the following entity types:

- Data sources
- Data mining tools
- Models and optimisation
- Data output

Data sources refer to systems that provide production-related data. The utilised message formats must cover at least measurement values from the production process, both actual as well as historic data. Further requirements may also appear, such as equipment states or process status information. For instance, the information about finishing a process step may trigger a scheduling action for the next step. A data source may be, for instance, a Distributed Control System (DCS).

Data mining tools are services for data mining operations. They are utilised to discover information in measured process data, so they consume historical data. The concrete data mining tools may be based on, e.g., WEKA, R or Python.

Modelling and optimisation tools help the management of production processes. They may provide, for instance, production schedules or other assistance.

Data output refers to exposing data to higher-level systems, such as the graphical user interfaces of operators. The actual data output is likely provided by calculation models and other optimisation modules.

These entity types are described in more detail in deliverable D3.1 Software Architecture Description for the Runtime System.

2.2 Message Serialisation

Due to their straightforward computational processing, text-based formats are utilised. For this, good candidates are JSON (JavaScript Object Notation) (ECMA-404, 2017) and XML (Extensible Markup Language) (Bray et al, 2008). For both JSON and XML, there is a solid support in software development tools. Still, it is important to recognise that JSON and XML alone are insufficient, as they do not specify any data structures but focus on general data serialisation. This does not reduce the applicability of JSON and XML, as various message format specifications have been designed on them.

2.3 Existing Message Format Specifications

The following table gives an overview of the existing specifications that are considered candidates for various APIs. Although multiple technologies are given, the technologies possibly cover only some aspects, which means that additional data structures may still be required. In the table, data analysis modules refer to any COCOP-related models or similar that perform production optimisation. Such modules may execute, for instance, simulation or data mining.

Table 2. Existing specifications considered for the APIs

Specification	Raw measurement values	Integration with legacy systems	Output from data analysis modules
OPC UA (OPC unified architecture specification part 1, 2015)	**Y**	**Y**	n
Observations and Measurements (Observations and Measurements, 2013)	**Y**	n	**Y**
ISA-88 (ANSI/ISA-88.00.01-2010, 2010) (B2MML, 2013)	n	**Y**	**Y**
ISA-95 (ANSI/ISA-95.00.01-2010, 2010) (B2MML, 2013)	n	**Y**	**Y**

OPC UA excels at equipment data presentation, but it also has serious limitations considering COCOP. OPC UA is not only a data delivery protocol but also a message format. If there are any existing data sources with an OPC UA interface, their integration is straightforward due to the solid tool support of OPC UA. OPC UA even supports historic data access (OPC unified architecture specification part 11, 2015), which is important when production is analysed over a time period. However, OPC UA has equipment data access as its main intention. That is, considering that the main scope of COCOP is manufacturing operations, the abstraction level of OPC UA is low. The OPC UA information models could be utilised as a guideline to help the design of message formats, but if other suitable formats exist, the information models may bring no additional value. In addition, an important question is whether it is violation to detach the OPC UA message formats from its data delivery medium. Finally, the most probable use case for OPC UA is to provide raw measurement data to be converted to other message formats. Then, the data would be delivered and utilised further.

Observations and Measurements (O&M) has some overlap with OPC UA, but it clearly has its place. O&M specifies suitable message formats for the delivery of measurement values. In contrast to OPC UA, O&M does not specify any particular communication protocol for the messages, such as a message bus or HTTP. Thus, O&M is a more lightweight alternative for the cases where solely message formats are needed. Despite its seemingly low-level scope - i.e., measurement values - O&M may be suitable even for the messaging related to production operations. Any related computational task, such as simulation, may be considered to indirectly generate measurement values. Thus, the original purpose of the message formats would not be violated. For measurement values, O&M has two major advantages. First, it specifies various measurement-related metadata items. For instance, there are fields to specify the utilised instrument or to distinguish the time a measurement result has been resolved from the actual measurement time. Second, O&M also provides multiple message types depending on what has been measured, including multi-field measurements or historical time series.

ISA-88 and ISA-95 define information structures for manufacturing operations management. ISA-88 is concerned with the control of individual production processes. Its main focus is batch process aspects, such as recipes and production records. In contrast, ISA-95 has a higher abstraction level. It aims at facilitating the integration of information systems related to manufacturing operations management. ISA-95 covers, for instance, scheduling, resourcing, production capabilities and personnel. There is also a specification to serialise the ISA-88 and ISA-95 data structures; B2MML (2013) (Business to Manufacturing Markup Language) specifies multiple XML structures for that purpose. Considering COCOP, B2MML provides production-related data structures to communicate using any protocol, such as a message bus or HTTP. In a lucky situation, some legacy systems may also support B2MML.

2.4 Physical Units in Messages

To present physical units in message, there are multiple specifications. Such a separate specification may be required, as the data structure standards may not define a way to encode physical units. Multiple ways are possible, and ambiguity as well as notational challenges may arise. For instance, "a" may stand for a year or an "are" (a unit of area). Furthermore, sometimes special characters may be required (such as the Greek letters or the '°' character indicating a degree), which may cause challenges in computational processing. Thus, to prevent any unit-related problems in message processing, there should be an agreement about a common way for encoding. To save work effort, existing specifications should be favoured. The following table compares the most remarkable measurement unit specifications that have been discovered. Each specification is briefly explained in the paragraphs after it.

Table 3. Standards and specification for measurement units

Specification	Suitability
UCUM (The Unified Code for Units of Measure) (Schadow & McDonald, 2014)	The best candidate found; extensive coverage, systematic coding approach
CML (Chemical Markup Language) (Chemical Markup Language, 2018)	Potential candidate; limited coverage
UNECE Codes for Units of Measure Used in International Trade (Recommendation No. 20, 2010)	Potential candidate; lacks intuition
UnitsML / UnitsDB (UnitsML Guidelines Version 1.0, 2011)	Unsuitable; no extensive, publicly available unit definitions provided

UCUM aims at the unambiguous representation of measurement units, and its goal is an extensive coverage of all measurement units that are currently relevant in various fields. The motivation of UCUM is the limited coverage and ambiguity of previously existing standards or specifications, and it has its emphasis on electronic communication. UCUM does not aim at an explicit specification of all the units in the world, because the number of units would be very large. Thus, to enable the encoding of any unit, UCUM only defines rules.

CML is especially focused on chemicals. The coverage of the specification is low compared to UCUM.

UNECE Codes have a good coverage. However, various codes are numeric, which makes it difficult to manually interpret some of them.

UnitsML is a schema specification to represent units. Unfortunately, the main focus is the schema rather than an actual unit specification. Related to UnitsML, NIST (National Institute of Standards and Technology) have developed UnitsDB, which actually specifies units, but it is not publicly available.

2.5 Messaging in Observations and Measurements

The Observations and Measurements standard (2013) specifies message formats for multiple observation types depending on the data contents. The following table provides an overview of the types considered most relevant. Still, even COCOP-specific message types may be specified if these types do not cover the needs of all scenarios. Still, such a custom message type may benefit from the metadata items specified for the existing types in Observations and Measurements.

Table 4. Observations and measurements

Type	Description
OM_Measurement	A measurement value; the actual value is a float, and the unit of measure is also given.
OM_CategoryObservation	The category of an object or entity. In the broad sense, a category may also be, for instance, the state of a production process.
OM_CountObservation	The count of some items. This may be, e.g., the number of items in a queue.
OM_TruthObservation	Whether a condition is true or not. Similar to a "boolean" value in various programming languages.
OM_ComplexObservation	An observation with multiple measured items. Each has a float value and a unit of measure.
OM_DiscreteTimeSeriesObservation	A series of sampling times and corresponding values. The type of the value may be any other measurement type.
OM_TemporalObservation	A time-related observation, such as the estimated finishing time of a process step.

In the following listing, there is an example of a measurement result structured as an Observations and Measurements message. The result represents a mass. The measurement was performed at 12.10 on a particular day, but the result was finished only later at 12.31. The result is 20.3 tons. There are also various metadata items indicating what has been measured and how.

- OM_Observation
    - @id: "batch_2018-02-05_14-30"
    - description: "The mass of a batch going to process step X1"
    - name: "Process step X1 batch mass"
    - type
        - @href: "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
    - phenomenonTime
        - TimeInstant
            - timePosition: "2018-02-05T12:10:13.00"

- resultTime
  - TimeInstant
    - timePosition: "2018-02-05T12:31:53.00"
- procedure
  - @href: "http://cocop/process_x1/massmeasurement"
- observedProperty
  - @href: "http://sweet.jpl.nasa.gov/2.3/propMass.owl#Mass"
- featureOfInterest
  - @href: "http://cocop/process_x1/batch"
- resultQuality
  - @href: "http://cocop/obsquality/good"
- result: "20.3"
  - type: "MeasureType"
  - @uom: "t"

2.6 Messaging in B2MML (ISA-88/ISA-95)

As the scope of B2MML is manufacturing operations, it may enclose production schedules, for instance. In the following listing, there is an example structure.

- ProductionSchedule
  - ProductionRequest
    - HierarchyScope
      - EquipmentID: "batch_cell_1"
      - EquipmentElementLevel: "ProcessCell"
    - StartTime: "2017-08-28T13:00:00"
    - EndTime: "2017-08-28T14:00:00"
  - ProductionRequest
    - HierarchyScope
      - EquipmentID: "batch_cell_2"
      - EquipmentElementLevel: "ProcessCell"
    - StartTime: "2017-08-28T13:30:00"
    - EndTime: "2017-08-28T14:30:00"

- ProductionRequest
  - HierarchyScope
    - EquipmentID: "batch_cell_3"
    - EquipmentElementLevel: "ProcessCell"
  - StartTime: "2017-08-28T14:00:00"
  - EndTime: "2017-08-28T15:00:00"

3 Conclusion

Considering the principles set in D3.1 Software Architecture Description for the Runtime System, this document has introduced technologies considered suitable or potential. Both communication protocols and message protocols are covered to enable interface implementation.

The following figure summarises the technologies and their positions regarding the functionality levels in production plants. The definitions of levels 1-4 are referred to according to ISA-95 (ANSI/ISA-95.00.01-2010, 2010, p. 21). Various specifications related to message formats and protocols are necessary for the platform. Some specifications are concerned with the integration of information systems, while others exchange equipment data. Due to its multi-purpose nature, the message bus spans all the levels. Observations and Measurements is possibly also utilised on all levels. Custom-made message formats are utilised where required; however, for the low-level communication with equipment, specifications such as OPC UA and Observations and Measurements are likely sufficient.

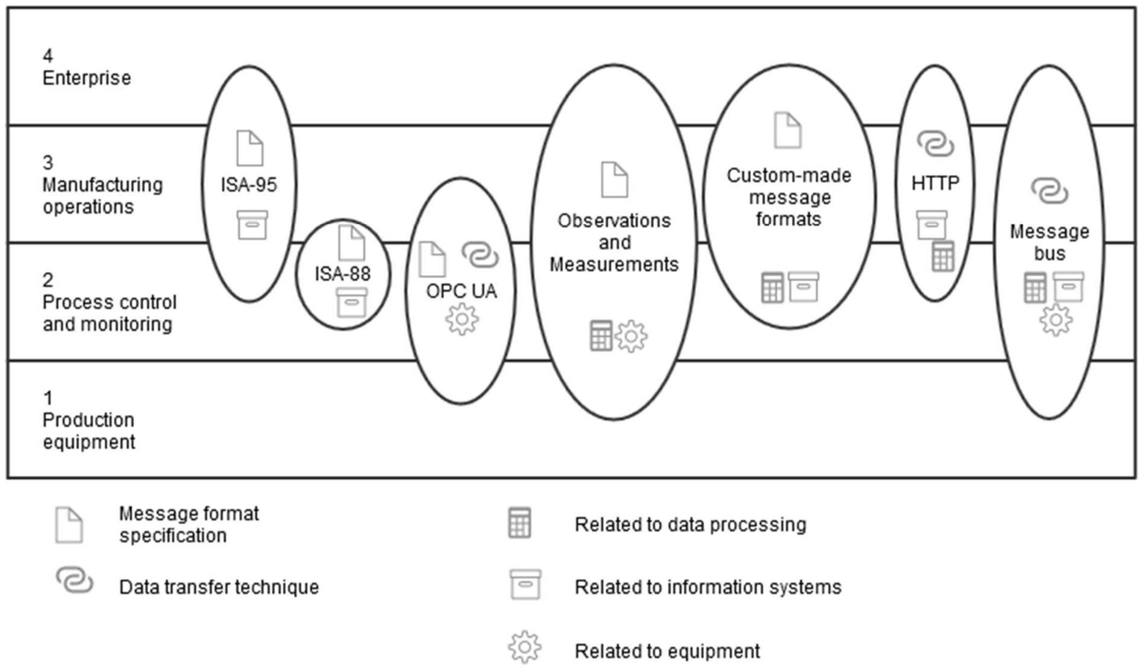
4
Enterprise
3
Manufacturing operations
2
Process control and monitoring
1
Production equipment
ISA-95
ISA-88
OPC UA
Observations and Measurements
Custom-made message formats
HTTP
Message bus
Message format specification
Data transfer technique
Related to data processing
Related to information systems
Related to equipment

Figure 3 Technologies positioned in functionality levels

The following table shows the function of each specification in the two-level stack of communication protocols and message formats. OPC UA defines both a data format and a communication protocol, but the other data formats may be freely combined with communication protocols according to what is appropriate.

Table 5 Functions of the specifications

	Communication protocol	Message format
Message bus (AMQP etc.)	x	
HTTP	x	
OPC UA	x	x
Observations and Measurements		x
ISA-88		x
ISA-95		x
Custom-made message formats		x

This document intentionally leaves room for further design. At this point, it remains partially unknown what the exact COCOP requirements are. Thus, the technologies mentioned in this document are solely candidates. Other technologies may also be utilised while not all the technologies mentioned here are necessarily utilised. The future document D3.7 Software architecture description for the runtime system (update) will give another view of the design of the COCOP concept.

References

Aiyagari, S., Arrott, M., Atwell, M., Brome, J., Conway, A., Godfrey, R., Greig, R., Hintjens, P., O'Hara, J., Radestock, M., Richardson, A., Ritchie, M., Sadjadi, S., Schloming, R., Shaw, S., Sustrik, M., Trieloff, C., van der Riet, K., and Vinoski, S., "AMQP. Advanced message queueing protocol. Version 0-9-1," http://www.amqp.org/specification/0-9-1/amqp-org-download (accessed 16.1.2018), 2008.

"ANSI/ISA-88.00.01-2010. Batch Control Part 1: Models and Terminology," International Society of Automation, 2010.

"ANSI/ISA-95.00.01-2010. IEC 62264-1 Mod. Enterprise-Control System Integration – Part 1: Models and Terminology," International Society of Automation, 2010.

"Apache Kafka," https://kafka.apache.org/ (accessed 16.1.2018), 2018.

"B2MML. Business to Manufacturing Markup Language," http://www.mesa.org/en/B2MML.asp (accessed: 19.1.2018), MESA International, 2013.

Bray, T., Paoli, J., Sperberg-McQueen, C., Mailer, Y., and Yergeau, F., "Extensible markup language (XML) 1.0 5th edition," https://www.w3.org/TR/2008/REC-xml-20081126/ (accessed 16.1.2018), W3C, 2008.

"Chemical Markup Language," http://www.xml-cml.org/ (accessed 16.1.2018), 2018.

"ECMA-404. The JSON data interchange syntax. 2nd edition," http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (accessed 16.1.2018), Ecma International, 2017.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., "RFC 2616. Hypertext Transfer Protocol – HTTP/1.1," https://tools.ietf.org/html/rfc2616 (accessed 24.1.2018), The Internet Society, 1999.

"Observations and Measurements. Version 2.0," http://www.opengeospatial.org/standards/om (accessed 19.1.2018), Open Geospatial Consortium, 2013.

"OPC unified architecture specification part 1: Overview and concepts. Release 1.03," https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/ (accessed 16.1.2018), OPC Foundation, 2015.

"OPC unified architecture specification part 11: Historical access. Release 1.03," https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-11-historical-access/ (accessed 24.1.2018), OPC Foundation, 2015.

"OPC unified architecture specification part 14: PubSub. Release candidate 1.04.24," https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/ (accessed 15.2.2018), OPC Foundation, 2017.

"Recommendation No. 20. Codes for Units of Measure Used in International Trade," http://tfig.unece.org/contents/recommendation-20.htm (accessed 16.1.2018), UNECE, 2010.

Schadow, G., and McDonald, C.J., "The Unified Code for Units of Measure," http://unitsofmeasure.org/ucum.html (accessed 16th Jan 2018), Regenstrief Institute, Inc. and the UCUM Organization, 2014.

"UnitsML Guidelines Version 1.0," https://unitsml.nist.gov/Presentations/oasis_flyer.pdf (accessed 16.1.2018), OASIS Open, 2011.

"ZeroMQ," http://zeromq.org/ (accessed 16.1.2018), 2018.