TECHNICAL REPORT

ISO/TR 16161

First edition
2019-06

Automation systems and integration — Use case of capability profiles for cooperation between manufacturing software units

Reference number
ISO/TR 16161:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The motivation for ISO 16100 stems from the industrial and economic environment, in particular:

a) a growing base of vendor-specific solutions;

b) user difficulties in applying standards;

c) the need to move to modular sets of system integration tools;

d) the recognition that application software and the expertise to apply that software are assets of the enterprise.

ISO 16100 is an International Standard for the computer-interpretable and human readable representation of a capability profile. Its goal is to provide a method to represent the capability of a manufacturing software unit (MSU) in manufacturing application software relative to its role throughout the life cycle of a manufacturing application, independent of a particular system architecture or implementation platform. This can lead to reduced production and information management costs to users and vendors/suppliers of manufacturing applications.

This document describes an application of ISO 16100. Manufacturing software agents, which are one type of MSU, achieve interoperation using capability profiles specified in ISO 16100.

This document describes message language and protocol for software agents to collaborate with each other to emerge systems function. Presenting the MSU capability profile defined in ISO 16100-3, the agents mutually recognize the capability of manufacturing activity and recognizable messages. Software agents that need manufacturing activities are called customers, and agents that provide manufacturing activities are called performers. Customers describe the request messages for manufacturing activities by the message language. Performers describe the report messages of the result of the manufacturing activities by the message language.

Agent Communication Language (ACL), proposed by the Foundation for Intelligent Physical Agents (FIPA), is a message language exchanged with multi agents, and the protocol that defines the sequence of messages is a protocol to which the framework of interaction protocol is applied and identifies the agent in order to use the ontology prescribed by FIPA. By contrast, this document describes the protocol and message language in which the software agent acting as a customer and the software agent acting as a performer interact in a one-to-one manner and each software agent is identified using a capability profile specified in ISO 16100-3. Therefore, the protocol and message language described in ACL and in this document are different.

Automation systems and integration — Use case of capability profiles for cooperation between manufacturing software units

1 Scope

This document describes an approach for using ISO 16100 to achieve cooperation between software agents by exchanging manufacturing software unit (MSU) capability profiles. The exchanged profiles among agents describe the manufacturing capabilities requested by the requester and to be fulfilled by the performer.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
customer
requester of the manufacturing activity

3.2
C-subsystem
manufacturing software unit requesting the manufacturing activity

3.3
performer
provider of the manufacturing activity

3.4
P-subsystem
manufacturing software unit providing the manufacturing activity

3.5
capability profile service provider
software that implements the capability profile interface

[SOURCE: ISO 16100-3:2005, 3.1.2]

3.6
service provider
entity that plays the role of *capability profile service provider* (3.5) and is responsible for preparing and delivering a pair of *customer* (3.1) and *performer* (3.3)

4 Abbreviated terms

MSU	Manufacturing Software Unit
UML	Unified Modelling Language
URI	Uniform Resource Identifier

5 Dialogue between C-subsystem and P-subsystem

The interaction that occurs between the orderer and the contractor is replaced by the dialogue between a customer and a performer. The dialogue between a customer and a performer can be represented by a sequence of actions associated with the order. The sequence of actions can be represented by the state transition diagram of dialogue. In a production system, the customer is the C-subsystem and the performer is the P-subsystem.

Figure 1 shows a dialogue that occurs between a C-subsystem and a P-subsystem. In Figure 1, "C: Request" is the operation of the request to a P-subsystem from a C-subsystem. "P: Promise" is an action promising that the P-subsystem has accepted the request to the C-subsystem. "P: Decline" is an action that the P-subsystem declines with respect to the request to the C-subsystem. Sometimes the P-subsystem offers a proposal. The C-subsystem accepts this proposal. The transition from state 2 to state 2’ shows a sequence of these actions. On the other hand, C-subsystem can offer another proposal for the proposal offered by the P-subsystem. The transition from state 2’ to state 2 shows a sequence of these actions.

It is assumed that the dialogue between the orderer and the contractor can be expressed by the state transition diagram of Figure 1, and messages are designed in accordance with this assumption. The following is a description of each state:

— State 1: initial state

— State 2: offer an order

— State 3: accept the order

— State 4: complete the order

— State 5: final state

— State 2': offer a proposal

— State 3', 2'', 3'': final state

Annex E gives an example of customization of a state transition diagram of dialogue.

Annex F proposes a solution for handling state transitions of dialogues, messages and capacity profile.

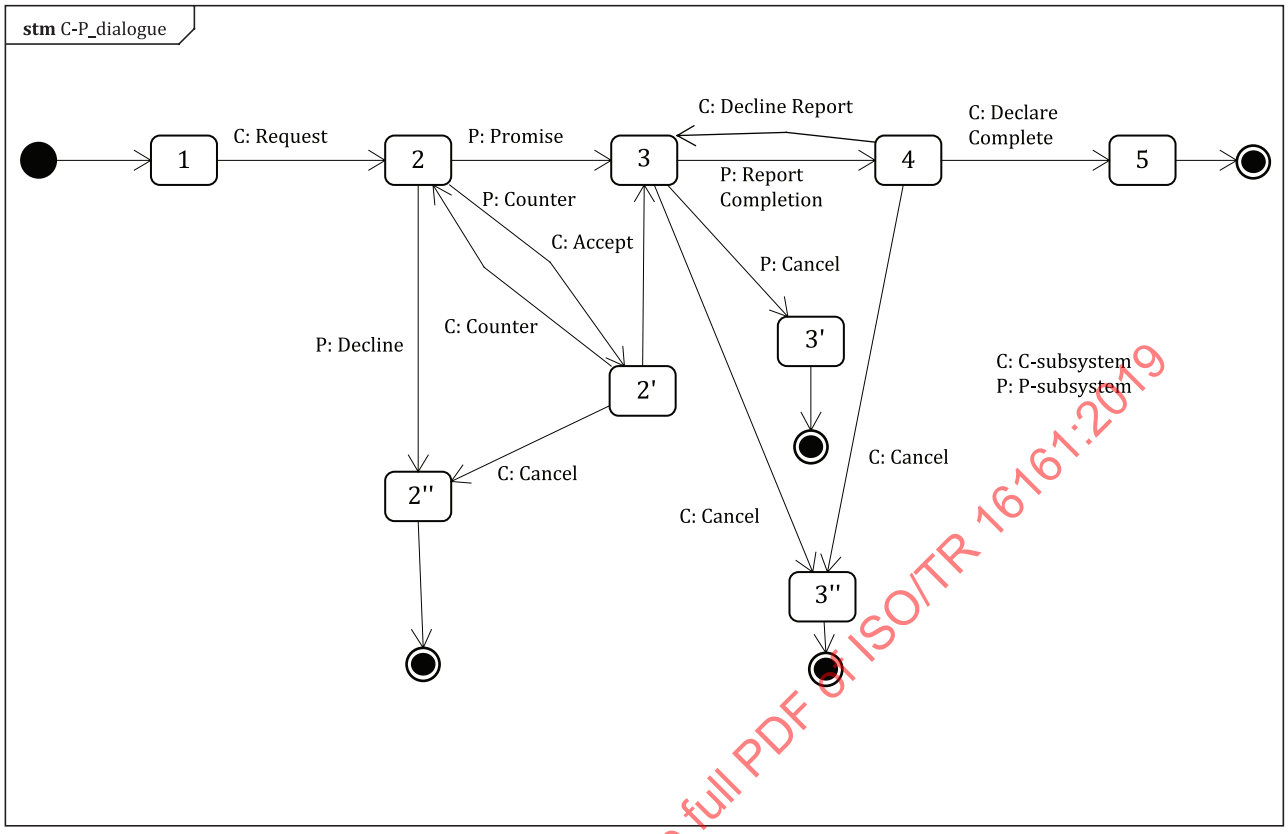

Figure 1 — State transition diagram of dialogue

6 Procedure until starting the dialogue between C-subsystem and P-subsystem

6.1 Overview

C-subsystem acquires the URIs of the P-subsystem that can perform a certain manufacturing activity from the service provider and the P-subsystem requested to perform the activity acquires the URIs of the C-subsystem which requests to perform the activity from the service provider. C-subsystem and P-subsystem start dialogue using the obtained URI respectively. When the C-subsystem sends the first Request message to the P-subsystem, the dialogue shown in Figure 1 starts and exchanges messages according to the procedure in Figure 1. This document assumes centralized control of bids by service providers which are defined in Clause 3. Other bidding processes, such as the distributed bidding process, are future works and are not described in this document.

6.2 Procedure for identifying the dialogue partner using capability profile

The procedure for identifying the dialogue partner and establishing the dialogue is as follows.

a) C-subsystem and P-subsystem register each capability profile in advance to the service provider. The details of the activities requested by C-subsystem are described in the capability profile of C-subsystem, and the details of the activities that P-subsystem can perform in the capability profile of P-subsystem are described.

b) The C-subsystem acquires the capability profile of the P-subsystem satisfying its own request from the service provider, and the P-subsystem acquires the capability profile of the requesting C-subsystem from the service provider. On matching of the capability profile, this procedure conforms to the procedure specified in the ISO 16100 series.

c) Before starting the dialogue in Figure 1, there are two cases of message exchanges between C-subsystem and P-subsystem. First case is that the C-subsystem sends a Notify message to the URI of the acquired P-subsystem, and the P-subsystem that received the Notify message immediately sends an Ask Response message to the C-subsystem. The other case is that the P-subsystem sends an Ask Response message even if there is no Notify message from the C-subsystem. The P-subsystem can repeatedly send the Ask Response message periodically or at an arbitrary timing until the response message is sent from the C-subsystem to the Ask Response message.

d) The dialogue begins when C-subsystem sends a Request message to P-subsystem. Subsequently, C-subsystem and C-subsystem exchange messages according to Figure 1.

The MSU realizes many different functions. The concern here is a communication function including how the C-subsystem and the P-subsystem establish a dialogue. Therefore, what is described here focuses on the communication function between the MSU as the role of the C-subsystem and the MSU as the role of the P-subsystem and the use of the capability profile in the communication. As Registration of capability profile in Figure 2 shows, before starting interaction between MSUs, each MSU needs to register each capability profile in advance with the service provider. The service provider determines the relationship between customer and performer between MSUs and confirms that both are connectable. This document assumes that the service provider has finished associating customer and performer between MSUs and that the service provider confirms that it can connect between the associated MSUs. The customer-performer model is illustrated in Annex C.

Figure 2 shows that the sequence diagram named 'C-P cooperation' consists of the sequence diagram named 'RegistationOfCapabilityProfiles', the sequence diagram named 'C-P dialogue', and the sequence diagram named 'C-P dialogue base'. The sequence diagram named 'RegistationOfCapabilityProfiles' shows that C-subsystem and P-subsystem register their capability profiles in the service provider beforehand. Since this document is an application example of capability profile in this document, details of 'RegistationOfCapabilityProfiles' and service provider are not explained. The sequence diagram named 'Prepare C-P dialogue' in Figure 2 consists of the sequence diagram named 'get capability profile' and the sequence diagram named 'send notify'. The first 'get capability profile' shows the interaction between MSU and the service provider. MSU acting as C-subsystem acquires capability profile of MSU acting as P-subsystem from the service provider. On the other hand, this P-subsystem obtains the capability profile of C-subsystem which is a partner of P-subsystem from the service provider. The second 'send notify' has two cases. First case is that when C-subsystem sends Notify message to P-subsystem and P-subsystem sends Ask Response to C-subsystem, the other case is when the P-subsystem sends Ask Response to the C-subsystem even if the P-subsystem does not receive the Notify message from the C-subsystem. 'Ask for Request message' in 'send notify' shows that P-subsystem can repeatedly send AskResponse until the C-subsystem responds with Request message.

The sequence diagram named C-P dialogue base in Figure 2 is a sequence diagram of the communication between the C-subsystem and the P-subsystem of Figure 1. As a result, only the C-subsystem needs to acquire the URI of P-subsystem, so parameter p is implementation-dependent and is not specified in this document, in particular. The sequence diagram of C-P dialogue base is described in Annex B.

Message exchanged between C-subsystem and P-subsystem is specified in Clause 7. Message communication between C-subsystem and P-subsystem is explained in more detail in Clause 8.

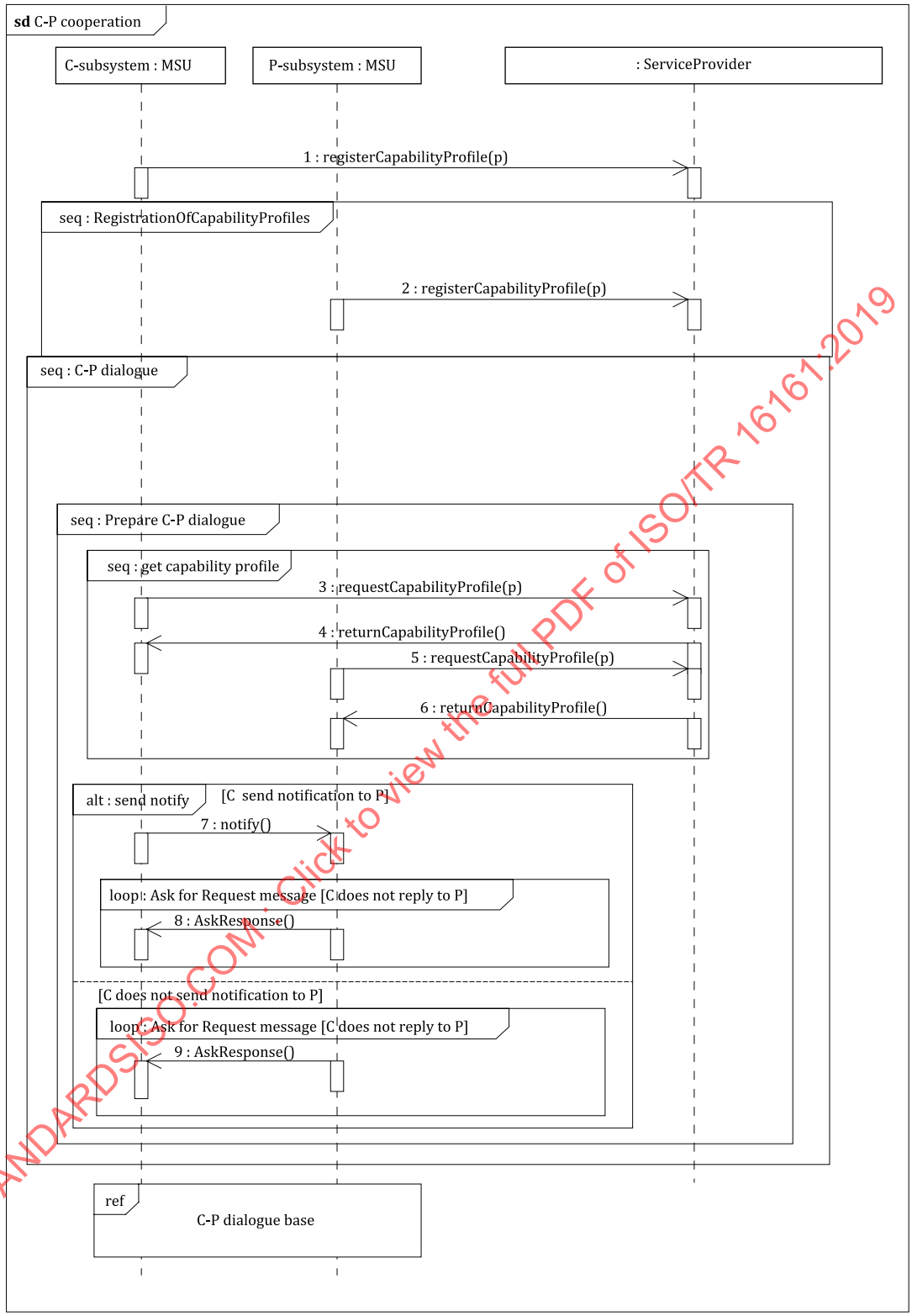

Figure 2 — Interaction overview diagram of dialogue

6.3 Capability profile

6.3.1 General

Using the capability profile, a MSU identifies the partner who dialogues with. Even if a C-subsystem sends Notify mess Annex B ages at first, every dialogue is started by Invitation message of a P-subsystem. Therefore, the communication channels of the C-subsystem need to be defined. In a case where the C-subsystem sends a Notify message, the address of P-subsystem is needed.

6.3.2 Description of specific part

Above information is described in the specific part of capability profile according to the template of the capability profile (see ISO 16100-3). The template of the specific part which added necessary elements in this document and an example of this specific part are shown in 6.4.

The XML tags added to the specific part are <Activity> and <Performatives>. These are used to describe the information needed by the adapters.

Tag <Activity> represents a manufacturing activity using a verb that the MSU provides capability. In Figure B.1 "Overall Structure of Production Management System", production activities for each domain implemented as MSU are represented by verbs such as Sell, Make, Buy, Bill, Pay, and so on. The <Channel> tag in the tag <InformationExchange> in the <Activity> section has information on communication between MSUs. The communication method is defined by the attribute name 'type' of this tag, and the URI is defined by the attribute name 'address'. The communication method will be explained in Clause 8.

Tag <Performatives> represents performative verbs (tag <Receive>) that can be receive and performative verbs (tag <Send>) of the MSU that can be sent, and tag <MessageFormat> gives information of data types in each message item.

6.4 Capability profile template and examples

In this clause, the template which added the necessary element in the specific part of the capability profile (see ISO 16100-3) is shown in 6.4.1, and an example of the capability profile applying this template is shown in 6.4.2.

6.4.1 Capability profile template

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CapabilityProfiling">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="type">
          <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="CapabilityProfile">
          <xs:complexType>
```



```
            <xs:sequence>
              <xs:element name="pkgtype">
                <xs:complexType>
                  <xs:attribute name="version" type="xs:string"
form="unqualified"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="Common" type="CommonPartType"/>
              <xs:element name="Specific" type="SpecificPartType"/>
            </xs:sequence>
            <xs:attribute name="date" type="xs:string" form="unqualified"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="CommonPartType">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Requirement">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" form="unqualified"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="MSU_Capability">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" form="unqualified"/>
```

```
    </xs:complexType>
  </xs:element>
</xs:choice>
<xs:sequence maxOccurs="unbounded">
  <xs:element name="ReferenceCapabilityClassStructure">
    <xs:complexType>
      <xs:attribute name="id" type="xs:string" form="unqualified"/>
      <xs:attribute name="name" type="xs:string" form="unqualified"/>
     <xs:attribute name="version" type="xs:string" form="unqualified"/>
      <xs:attribute name="url" type="xs:string" form="unqualified"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="TemplateID">
    <xs:complexType>
      <xs:attribute name="ID" type="xs:string" form="unqualified"/>
    </xs:complexType>
  </xs:element>
</xs:sequence>
<xs:element name="Version">
  <xs:complexType>
    <xs:attribute name="major" type="xs:string" form="unqualified"/>
    <xs:attribute name="minor" type="xs:string" form="unqualified"/>
  </xs:complexType>
</xs:element>
<xs:element name="Owner">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="0"/>
      <xs:element name="street" type="xs:string" minOccurs="0"/>
      <xs:element name="city" type="xs:string" minOccurs="0"/>
      <xs:element name="zip" type="xs:string" minOccurs="0"/>
      <xs:element name="state" type="xs:string" minOccurs="0"/>
      <xs:element name="country" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="comment" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ComputingFacilities" minOccurs="0"
maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
         <xs:element name="Processor0" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="type" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="OperatingSystem0" minOccurs="0"
maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="type" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
          <xs:element name="Language" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="name" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="Memory" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="size" type="xs:string" form="unqualified"/>
             <xs:attribute name="unit" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
          <xs:element name="DiskSpace" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="size" type="xs:string" form="unqualified"/>
```

```
            <xs:attribute name="unit" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="type" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Performance" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
      <xs:attribute name="ElapsedTime" type="xs:string" form="unqualified"/>
          <xs:attribute name="TransactionsPerUnitTime" type="xs:string"
form="unqualified"/>
        </xs:complexType>
      </xs:element>
    <xs:element name="ReliabilityData" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
           <xs:element name="UsageHistory" type="xs:string" minOccurs="0"/>
          <xs:element name="Shipments" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
           <xs:attribute name="number" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="IntendedSafetyIntegrity" minOccurs="0"
maxOccurs="unbounded">
              <xs:complexType>
            <xs:attribute name="level" type="xs:string" form="unqualified"/>
              </xs:complexType>
            </xs:element>
       <xs:element name="Certification" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
             <xs:attribute name="no1" type="xs:string" form="unqualified"/>
              </xs:complexType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SupportPolicy" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="index" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="PriceData" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="invest" type="xs:string" form="unqualified"/>
          <xs:attribute name="annualSupport" type="xs:string"
form="unqualified"/>
          <xs:attribute name="unit" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SpecificPartType">
    <xs:sequence>
      <xs:element name="Activity" type="Activity" maxOccurs="unbounded"/>
      <xs:element name="Performatives" type="Performatives" />
      <xs:element name="MessageFormat" type="MessageFormat" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Activity">
    <xs:sequence>
      <xs:element name="InformationExchange" type="InformationExchange"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" form="unqualified"/>
```

```
    <xs:attribute name="name" type="xs:string" form="unqualified"/>
  </xs:complexType>
  <xs:complexType name="InformationExchange">
    <xs:sequence>
      <xs:element name="BasicProtocol">
        <xs:complexType>
          <xs:attribute name="id" type="xs:string" form="unqualified"/>
          <xs:attribute name="type" type="ProtocolType" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="ApplicationProtocol">
        <xs:complexType>
          <xs:attribute name="id" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="TalkTo">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Method" type="Method" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:string" form="unqualified"/>
          <xs:attribute name="address" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="CallbackTo" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Method" type="Method" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:string" form="unqualified"/>
          <xs:attribute name="address" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
```

```
      <xs:element name="ContentEditor" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Method">
    <xs:attribute name="type" form="unqualified">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="GET" />
          <xs:enumeration value="POST" />
          <xs:enumeration value="PUT" />
          <xs:enumeration value="DELETE" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="name" type="xs:string" form="unqualified"/>
  </xs:complexType>
  <xs:simpleType name="ProtocolType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PUSH" />
      <xs:enumeration value="PULL" />
      <xs:enumeration value="NOTIFY" />
      <xs:enumeration value="CALLBACK" />
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="Performatives">
    <xs:choice maxOccurs="unbounded">
      <xs:element name="Receive">
        <xs:complexType>
```

```
          <xs:attribute name="name" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Send">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="MessageFormat">
    <xs:sequence>
      <xs:element name="Data" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string" form="unqualified"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

6.4.2 Capability profile example

The following example is the capability profile of the MSU (C-subsystem) which perform the Sell activity. This example is limited to the elements necessary for the dialogue with the MSU acting as the P-subsystem.

```
<?xml version="1.0" encoding="UTF-8"?>
<CapabilityProfiling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <type id="MSU_Profile"/>
  <CapabilityProfile date="2017-08-10">
    <pkgtype version="1.0" />
    <Common>
      <MSU_Capability>
        <ID>Buy/Make Subsystem</ID>
      </MSU_Capability>
      <ReferenceCapabilityClassStructure />
```



```
    <TemplateID />
    <Version />
    <Owner>
      <name>APSOM MESX-JP</name>
      <country>Japan</country>
    </Owner>
  </Common>
  <Specific>
    <Activity id="Sell" name="SELL">
      <!-- Pull channel from Sell-function as C-subsystem -->
      <InformationExchange>
        <BasicProtocol id="REST" type="PULL" />
        <ApplicationProtocol id="C-subsystem-P-subsystem" />
        <TalkTo id="Sell" address="http:// URL of P-subsystem /ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </TalkTo>
        <CallbackTo id="Sell" address="http://localhost/ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </CallbackTo>
       <ContentEditor name="IHCL.message.editor.DefaultContentEditor" />
      </InformationExchange>
      <!-- Push channel to Sell-function as C-subsystem -->
      <InformationExchange>
        <BasicProtocol id="REST" type="PUSH"/>
        <ApplicationProtocol id="C-subsystem-P-subsystem" />
        <TalkTo id="Sell" address="http:// URL of P-subsystem /ihcl/">
          <Method type="POST" name="ihcladaptor" />
        </TalkTo>
      </InformationExchange>
    </Activity>
    <Activity id="Notify" name="CallFor">
      <!-- Notify channel to P-subsystem Do -->
      <InformationExchange>
```

```
          <BasicProtocol id="REST" type="PUSH" />
          <ApplicationProtocol id="C-subsystem-P-subsystem" />
         <TalkTo id="Do" address="http:// address of  do subsystem /ihcl/">
            <Method type="POST" name="IHCLAdaptorDefaultServlet" />
          </TalkTo>
         <ContentEditor name="IHCL.message.editor.DefaultContentEditor" />
        </InformationExchange>
      </Activity>
      <Performatives>
        <Receive name="Request" />
        <Send name="Promise" />
        <Send name="Decline" />
        <Send name="ReportCompletion" />
        <Receive name="DeclineReport" />
        <Receive name="DeclareComplete" />
      </Performatives>
      <MessageFormat>
        <Data name="o-id" />      <!--  Order identifier -->
        <Data name="who" />       <!--  Orderer -->
        <Data name="what" />      <!--  Order item -->
        <Data name="spec" />      <!--  Specification-->
        <Data name="how-many" /> <!-- quantity -->
        <Data name="when_by" />  <!--  Due date -->
        <Data name="where_to" /> <!--  Destination -->
        <Data name="option" />    <!--  Extra information-->
      </MessageFormat>
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

7 Message

7.1 Structure of Message

Message is specialized as either a ControlMessage or a PerformativeMessage. PerformativeMessage causes the state transition of the dialogue. ControlMessage does not change the state of dialogue. Figure 3 shows structure of Message.

There are standards such as IEC 62264-5 and OAGiS that define transactions or messages relating to information exchange between applications. And verbs play an important role in both IEC 62264-5 and OAGiS, and in this point it is the same as the message of this document. On the other hand, messages of this document are used for the dialogue between applications applying capability profiles, and it is based on a framework different from messages of OAGiS and transactions of IEC 62264-5. Since verbs of transaction of IEC 62264-5 and verbs of message of OAGiS are the same except for the verb 'Notify', Annex G shows the mapping table between verbs of this message and verbs of OAGiS.

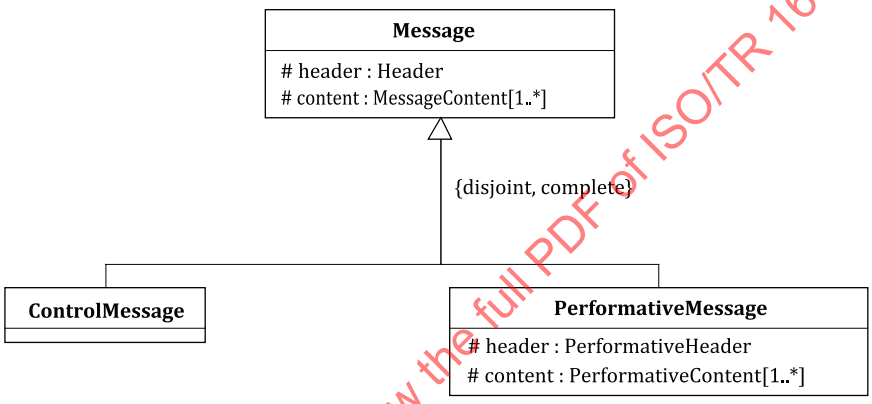

Figure 3 — Structure of Message

Message consists of an attribute header and attributes content. The header is an attribute with type Header. The content is an attribute of type MessageContent with multiplicity 1..*.

Header is specialized as either a ControlHeader or a PerformativeHeader. Figure 4 shows structure of Header.

Header consists of an attribute verbtype with type String, an attribute verb with type String and an attribute named clause. The clause is an attribute of type Clause with multiplicity 1..*. Whether the Header instance is an instance of ControlHeader or an instance of PerformativeHeader depends on the value of the attribute verbtype. Value of the attribute verbtype is limited to “controlVerb” or “performative”. When the value of verbtype is “controlVerb”, it is an instance of ControlHeader. When the value of verbtype is “performative”, it is an instance of PerformativeHeader. Clause consists of an attribute name with type String and an attribute value with type String.

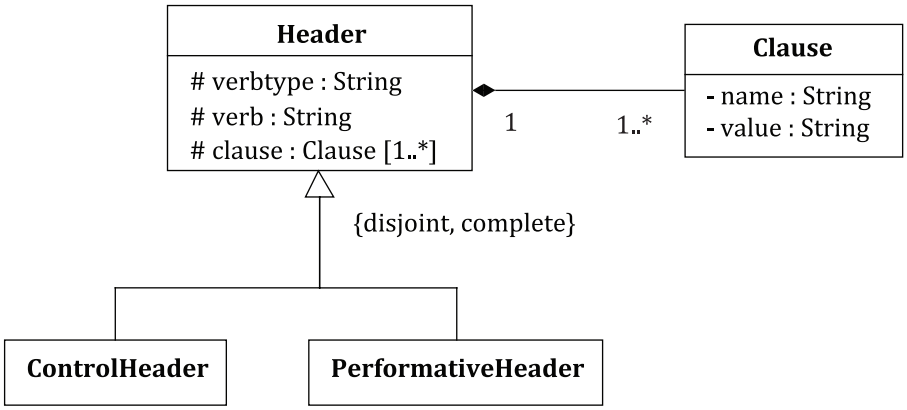

Figure 4 — Structure of Header

Table 1 — Value of Clause

name	value
sender	Identifier of the sender.
receiver	identifier of recipient
time-stamp	timestamp when sending message
message-id	message identifier
in-reply-to	original message identifier
language	"zzzz"

MessageContent is specialized as either a ControlContent or a PerformativeContent. Figure 5 shows structure of MessageContent.

Control Content has an attribute message-content with type String. Performative Content has an attribute message-content with type Performative Message Content.

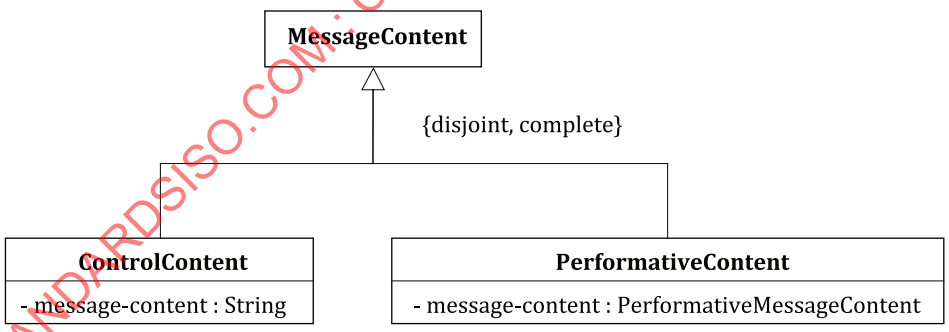

Figure 5 — Structure of MessageContent

PerformativeMessageContent consist of an attributes order with type String , attribute ordereditems with type NameValuePair with multiplicity 1..*, and an attribute checker of with type NameValuePair with multiplicity 0..2. Figure 6 shows structure of PerformativeMessageContent.

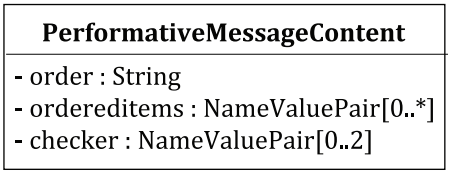

Figure 6 — Structure of PerformativeMessageContent

An instance of Order Attributes consists of a value with type String. An instance of Ordered Items consists of one or more instance of Ordered Item. An instance of Checkers consists of one or more instance of Checker.

NamedValuePair is specialized as ether a NumberOfPieces, Quantity, StringValue or CollectionValue. Quantity consists of two attributes, an attribute name value and an attribute name measure. The attribute value is a double type and the attribute measure is a String type. Quantity is used to represent physical quantities such as length and weight. CollectionValue has an attribute value with type NamedValuePair with multiplicity 1..*. CollectionValue is used to express nested instances of NameValuePair.

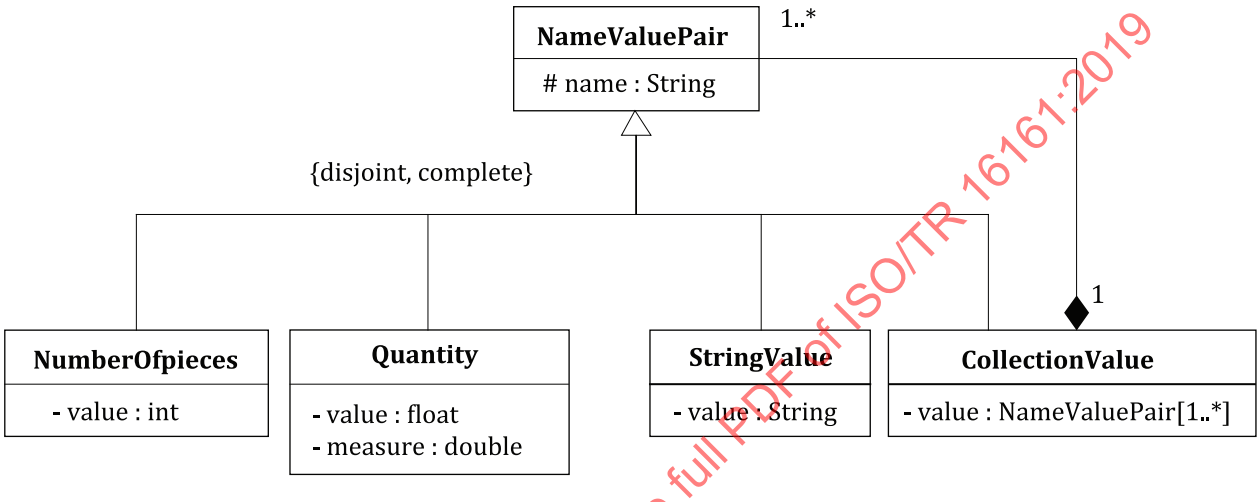

Figure 7 — Structure of NameValuePair

7.2 Control Message

The control message has the following message.

— Notify message.

— Ask Response message.

— Warn message.

In the following, each of these messages will be explained by using examples.

7.2.1 Notify message

A C-subsystem notifies the P-subsystem that the C-subsystem is ready to place an order for the P-subsystem. When the C-subsystem sends this message to the P-subsystem, it triggers the P-subsystem to send an Invitation message.

Message is an instance of ControlMessage. The attribute header has an instance of ControlHeader. The attribute content has an instance of ControlContent. When C-subsystem has a requirement for P-subsystem, C-subsystem sends to P-subsystem this message. The reason why this message is optional is explained in the Ask Response message.

The following is the attribute value of instance of ControlHeader.

— verbtype="controlVerb"

— verb="Notify"

— clause = (("sender","Cxxxx"),("receiver", "piny"),....,("language","zzzz"))

The following is the attribute value of instance of ControlContent.

— message-content=""

7.2.2 Ask Response message

This message is used for both message returned as a response to Notify message and message prompting response when waiting for a response from C-subsystem. P-subsystem receives a Notify message from the C-subsystem and the P-subsystem sends an Ask Response message. The C-subsystem responds to the Ask Response message and replies to the P-subsystem that sent this message a message such as Request. As described in 6.2, when P-subsystem asks C-subsystem for response of C: Request message, P-subsystem can send Ask Response message repeatedly to C-subsystem. In addition to this case, when P-subsystem asks C: Counter, C: Cancel or C: DeclareComplete message to C-subsystem, P-subsystem can send an Ask Response message repeatedly to C-subsystem until the desired message is returned.

Message is an instance of ControlMessage. The attribute header has an instance of ControlHeader. The attribute content has an instance of ControlContent. P-subsystem sends to C-subsystem this message.

The following is the attribute value of instance of ControlHeader.

— verbtype="controlVerb"

— verb="AskResponse"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of ControlContent.

— message-content=select (*) where ('ConditionStatement')

ConditionStatement is "ID=RequestIdentifier"

RequestIdentifier is Identifier of Request or "NEW"

EXAMPLE 1 P-subsystem already received a request from C-subsystem, the request identified "RE12345".

ConditionStatement is "ID=RE12345".

This conditonStatement means that P-subsystem asks to C-subsystem resending the request.

EXAMPLE 2 P-subsystem received a request after receiving another request.

ConditionStatement is "ID=NEW"

P-subsystem identifies the message received from C-subsystem by specifying the time-stamp received last time. The P-subsystem sends a reply to this message to C-subsystem.

7.2.3 Warn message

If the recipient of the message detects something abnormal during the dialogue, it sends a warning message to that message sender.

Message is an instance of ControlMessage. The attribute header has an instance of ControlHeader. The attribute content has an instance of ControlContent. When P-subsystem detects an abnormal state, the P-subsystem sends to C-subsystem this message. When the C-subsystem detects an abnormal state, the C-subsystem sends to the P-subsystem this message.

The following is the attribute value of instance of ControlHeader.

— verbtype="controlVerb"

— verb="Warn"

— clause = (("sender","Cxxxx"),("receiver","piny"),...,("language","zzzz"))

The following is the attribute value of instance of ControlContent.

— message-content=error at 01000001 Request

01000001 is message-id which is the identifier of message which has some abnormal at the Request message.

The error wording is one of the following words.

— "Protocol Sequence Error": Unexpected message received, the message cannot be accepted at current state of dialogue.

— or "Message Duplicated": Received same message

— or "NoR Unmatched": Not match requested count

— or "Checksum Unmatch": Checksum is not same value.

7.3 PerformativeMessage

PerformativeMessage is classified into 9 types. Each type corresponds to an action of Figure 1. Table 2 shows these correspondences.

Table 2 — Classified type of PerformativeMessage.

Type of Message	Action	Description
Request message	C:Request	C-subsystem sends a Request message which contains orders to P-subsystem.
Promise message	P:Promise	P-subsystem sends the Promise message to C-subsystem.
Counter message	P:Counter, C:Counter	P-subsystem sends a counteroffer to C-subsystem. C-subsystem sends a counteroffer to P-subsystem.
Accept message	C:Accept	C-subsystem accepts a counteroffer from P-subsystem.
Decline message	P:Decline	P-subsystem declines C: Request or C: Counter from C-subsystem.
Cancel message	C:Cancel, P:Cancel	C-subsystem cancels an order issued by the C-subsystem. P-subsystem cancels an order issued by the C-subsystem.
Report Completion message	P:Report Completion	P-subsystem sends a Report Completion message to C-subsystem
Declare Complete message	C:Declare Complete	C-subsystem sends Declare Complete message when C-subsystem accepts Report Completion message from P-subsystem.
Decline Report message	C:Decline Report	C-subsystem sends Decline Report message when C-subsystem refuses Report Completion message from P-subsystem.

Table 3 — Name list of ordereditems of PerformativeMessageContent

Name	Description
o-id	identifier of an order
who	Orderer, party or organization
what	Identifier of item or account name
spec	Specification of ordered item, CollectionValue

Table 3 *(continued)*

Name	Description
serialno	Identifier of ordered item
how many	Ordering quantity (number of pieces or Quantity representing physical quantity)
how much	Price, amount
when_by	Desired delivery date and time or completion date, date and time
when_at	Desired start date and time
where_from	Shipment location, place or party
where_to	Receiving point, place or party
whom_incharge	Department in charge or implementation process, organization
option	Remarks, text

Table 4 — Relationship between ordered items and Performative Message

Name of Ordered item	M/O	Request message	Promise message	Counter message	Accept message	Decline message	Cancel message	Report Completion message	Declare Complete message	Decline Report message
o-id	M	M	M	M	M	M	M	M	M	M
Who	M	M	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
what	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
Spec	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
serialno	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
how many	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
how much	M	M	M	M	N/A	N/A	N/A	M	N/A	N/A
when_by	O	O	O	O	N/A	N/A	N/A	M	N/A	N/A
when_at	O	O	O	O	N/A	N/A	N/A	M	N/A	N/A
where_from	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
where_to	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
whom_incharge	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
option	O	O	O	O	N/A	N/A	N/A	O	N/A	N/A
M: mandatory, O: option, N/A: not applicable "when_by" and "when_at" are exclusive.										

This document does not specify for the proper use of numbers and quantities. Refer to the product definition information (IEC 62264-3).

7.3.1 Request message

C-subsystem places orders in the P-subsystem using this message type as a response to a request to wait for a response issued by the P-subsystem. A message of this type, Request, can contain multiple orders in one message. During the execution of the order, progress is measured by order. When the event of the dialogue becomes C: Declare Complete, C: Cancel, P: Cancel, P: Decline, the progress management is terminated.

— verbtype="performative"

— verb="Request"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

— order ="Buy"

The value of order is corresponding to the identifier of activity defined in the capability profile. The owner of the production system (Owner) defines the value of order and the identifier of activity defined in the capability profile. Examples of frequently used order's values are shown in Annex B.

For example, the order's value "Make" means that the item ordered is completed by the desired date according to the ordered specifications. The order's value “Ship" means that the item ordered has arrived at the specified addressee from the specified warehouse by the desired date.

— ordereditems=(("o-id","#0004"),("who","CUST01"),("what","screw"),...,("when_by","2017-10-02"),... ,("whom_incharge","ABC"),(("option","option is ..."))

— The capability profile defines which item in ordereditems to include in this message. The name "what" requests the identifier of item as its value.

— checker=(("number-of-records",1), ("check-sum",13))

In this example, it is shown that the number of records and the checksum value are confirmed.

Optional items can be omitted. Interpretation of the default value is separately determined by the application system.

7.3.2 Promise message

The P-subsystem receives the Request message from the C-subsystem. When the P-subsystem judges that the specifications, number, delivery date etc. of the item are acceptable level, the P-subsystem returns a "promise (P: Promise)" message. This causes a transition from state 2 to 3 in Figure 1. Whether the P-subsystem returns an estimate of price and/or an estimated delivery date for the order in this message or not will be determined separately in the overall system.

The ordereditems of the Promise message contain the order identification and only the difference from the ordereditem of the Request message. If all of the ordereditems in the Promise message corresponding to the ordereditem in the Request message is the same, only the o-id is the ordereditem , order identifier, in the Promise message.

— verbtype="performative"

— verb="Promise"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

— order="Buy"

— ordereditems=(("o-id","#0004"),)

It shows the case where the ordereditem of the Request message and all the corresponding ordereditems of the Promise message are the same.

— checker=(("number-of-records",1), ("check-sum",13))

In this example, it is shown that the number of records and the checksum value are confirmed.

7.3.3 Counter message(P: Counter)

Although the P-subsystem received the request message from the C-subsystem, the specification, number or quantity of items, delivery date, amount, etc. can exceed the allowable range. At this time the P-subsystem can present a counteroffer to C-subsystem. This causes a transition from state 2 to 6 in Figure 1.

— verbtype="performative"

— verb="Counter"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

— order ="Buy"

— ordereditems=(("o-id","#0004"),("when_by","2017-10-10"))

It shows the case that the P-subsystem presents a counteroffer to change the delivery date.

— checker=(("number-of-records",1), ("check-sum",13))

7.3.4 Accept message

When the C-subsystem accepts the Counter message returned by the P-subsystem, the C-subsystem returns Accept message to the P-subsystem. This causes a transition from state 2' to 3 in Figure 1.

— verbtype="performative"

— verb="Accept"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

— order ="Buy"

— ordereditems=(("o-id","#0004"))

— checker=(("number-of-records",1), ("check-sum",13))

7.3.5 Cancel message (against Counter C: Cancel)

When the C-subsystem does not accept the Counter message which returned by the P-subsystem, the C-subsystem returns Cancel message to the P-subsystem. This causes a transition from state 2' to 2" in Figure 1.

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="Cancel"

clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order ="Buy"

— ordereditems=(("o-id","#0004"))

7.3.6 Counter message(C: Counter)

When C-subsystem can not accept the counteroffer (P: Counter) sent by the P-subsystem, C-subsystem's reply is either cancelling the Request (C: Cancel) or counteroffer (C: Counter) to the P-subsystem's counteroffer (P: Counter). When the C-subsystem decides that there is a possibility of being able to contract with the P-subsystem, the C-subsystem responds to the counteroffer (C: Counter). The counteroffer of the C-subsystem (C: Counter) is a modification of any one (or all) of items, specifications, delivery date, quantity, among P-subsystems' counteroffer (P: Counter). This causes a transition from state 2' to 2 in Figure 1. During this time, the state of Request is pending.

verbtype="performative"

verb="Counter"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

— order ="Buy"

— ordereditems=(("o-id","#0004"),("when_by","2017-10-09"))

— It shows the case that the C-subsystem presents a counteroffer to change the delivery date.

— checker=(("number-of-records",1), ("check-sum",13))

7.3.7 Decline message

When the P-subsystem rejects C: Request or C: Counter from C-subsystem, the P-subsystem returns a Decline message. This causes a transition from state 2 to 3 in Figure 1.

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="Decline"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— Message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order="Buy"

— ordereditems=((“o-id","#0004"))

7.3.8 Cancel message (P: Cancel)

The P-subsystem basically sends a ReportCompletion message when stopping a promised order midway for certain reasons. The P-subsystem can also send a "P: Cancel" message. This is limited to a one-sided and special situation which interrupts dialogue with C-subsystem. This causes a transition from state 3 to 3’ in Figure 1.

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="Cancel"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order="Buy"

— ordereditems=((“o-id","#0004"))

7.3.9 Cancel message (C: Cancel)

The C-subsystem sends a Cancel message to the P-subsystem when stopping the orders, that are set out in the middle, or cancelling the completed order due to the circumstances of C-subsystem. Handling of

products that are finished until the middle is determined by the outside of the system. This causes a transition from state 3 to 9 or from state 4 to 9 in Figure 1.

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="Cancel"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy")...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order="Buy"

— ordereditems=(("o-id","#0004"))

7.3.10 Report Completion message

When the P-subsystem finishes or stops work indicated by the Request, the P-subsystem sends a Report Completion message to the C-subsystem who is the contractor of this work. This causes a transition from state 3 to 4 in Figure 1. If the P-subsystem decides to terminate, the P-subsystem sends a RepotCmpletion message even if the outcome does not satisfy the quantity, the delivery date, and even the specification. The C-subsystem decides whether or not it is sufficient.

verbtype="performative"

verb="ReportCompletion"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy")...,("language","zzzz"))

— order="Buy"

— ordereditems=(("o-id","#0004"),...,("how_many","1000"),("how_much","$200,000")
,("when_by","2017-10-09"))

An ordereditem with the names "o-id", "how_many", "how_much" and "when_by" is mandatory for the ordereditems.

— checker=(("number-of-records",1), ("check-sum",13))

7.3.11 DeclareComplete message

The C-subsystem decides whether to return an accepting message(DeclareComplete message) or a refusal message(DeclineReport message) to the ReportCompletion message sent from the P-subsystem. Even if the quantity is insufficient or the delivery date is delayed, the C-subsystem returns a message DeclareComplete message that C-subsystem accepts. This causes a transition from state 4 to 5 in Figure 1.

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="DeclareComplete"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order="Buy"

— ordereditems=(("o-id","#0004"))

7.3.12 Decline Report message

If the C-subsystem refuses to ReportComplete message, send the DeclineReport message to the P-subsystem. This causes a transition from state 3 to 4 in Figure 1. If the C-subsystem refuses to ReportComplete message, send the DeclineReport message to the P-subsystem. Decide whether to resume work outside the system (business judgment, etc.). The determination of whether to resume the work is the judgment of the outside of the system (such as a business decision).

Message is an instance of PerformativeMessage. Header is an instance of PerformativeHeader. The following is the attribute value of instance of PerformativeHeader.

— verbtype="performative"

— verb="DeclineReport"

— clause = (("sender","Cxxxx"),("receiver","Pyyyy"),...,("language","zzzz"))

The following is the attribute value of instance of PerformativeContent.

— message-content has an instance of PerformativeMessageContent.

The following is the attribute value of instance of PerformativeMessageContent.

— order="Buy"

— ordereditems=(("o-id","#0004"))

8 Communication protocol between C-subsystem and P-subsystem

8.1 General

The dialogue function between C-subsystem and P-subsystem described in Clauses 6 and 7 was realized. Annex G outlines the implementation example. In advance of explaining this realization example, this section describes the interface between C-subsystem and P-subsystem; message transfer method adopted and message communication method.

8.2 Interface between C-subsystem and P-subsystem

Clauses 6 and 7 state that the direction of the dialogue message is determined by the message type. The direction of C-subsystem is inbound from P-subsystem and the direction of P-subsystem is called outbound from C-subsystem. Interactions messages between C-subsystem and P-subsystem can be classified as follows and belong to one of these.

a) P-subsystem requests a response from C-subsystem (inbound, outbound)

b) A message (inbound) that the P-subsystem sends to C-subsystem only

c) Messages just sent to C-subsystem by P-subsystem (outbound)

For the Notify message that the C-subsystem sends to the P-subsystem, the P-subsystem has a Notify interface that accepts requests from the C-subsystem and the others have interfaces that accept requests

from the P-subsystem Have. The operation that P-subsystem sends to C-subsystem is Push, and the action that P-subsystem requests response from C-subsystem is Pull. Notify message. The relationship between push and pull interfaces and C-subsystem and P-subsystem other than the interface is shown in Figure 8. Examples of messages are given in Annex D.

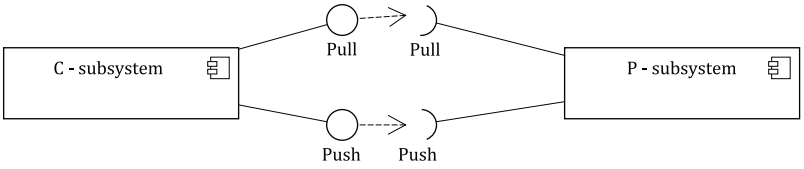

Figure 8 — Interface between C-subsystem and P-subsystem without Notify interface

8.3 Message communication in a dialogue between C-subsystem and P-subsystem

HTTP 1.1 was adopted as a message transfer protocol. The reason is that the architecture style of RESTfull was kept in mind and it is convenient for implementation. When the dialogue message passes through the interface between C-subsystem and P-subsystem, the path through which this message passes is called channel. For example, the path of a message passing through a pull interface is a push channel. In the dialogue, messages are exchanged asynchronously, so the message that passed the pull channel is received by the C-subsystem and then the pull channel is disconnected. Since there is no path through which the response message to this message passes, a callback interface to which the C-subsystem responds is added.

Table 5 shows the correspondence between the interface name, the channel name which is the route of the message passing through this interface, and the type attribute value of the Channel tag of the capability profile. The relation between the direction of the dialogue message and the channel is that the inbound message passes through the push channel or pull channel and the outbound message passes through the notify channel or callback channel. Table 6 shows the relationship between message type and action and channel.

Table 5 — Correspondence between interface name and channel name

interface name	channel name	type in Channel Tag
Push	push channel	push
Pull	pull channel	pull
Notify	notify channel	notify
Callback	callback channel	callback

Table 6 — Relationship between message type and channel

Type of Message	Action	outbound	inbound
Notify message	Notify	Notify channel	
Ask Response message	AskResponse		pull channel
Request message	C: Request	callback channel	
Promise message	P: Promise		push channel
Counter message	P: Counter		pull channel
	C: Counter	callback channel	
Accept message	C: Accept	callback channel	
Decline message	P: Decline		push channel
Cancel message	C: Cancel	callback channel	
	P: Cancel		push channel

Table 6 *(continued)*

Type of Message	Action	outbound	inbound
Report Completion message	P: Report Completion		pull channel
Declare Complete message	C: Declare Complete	callback channel	
Decline Report message	C: Decline Report	callback channel	

Annex A
(informative)

Message definition in JSON schema

This annex describes the message format in text form; the message format is expressed by Json (JavaScript Object Notation) schema. The rules of the sequence of the message are described in Clause 7. This document assumes that a MSU exchange messages defined by Json schema.

```
{
// Json Schema of messages in this document
  "$schema": "http://json-schema.org/draft-04/schema#",


  "definitions": {
     // simple types
     "integertype" : { "type" : "integer" },
     "numbertype" : { "type" : "number" },
     "datetype" : { " type" : "string",
            "format": "date",
            "description": "ISO8601/RFC3339 formats for date",
     },
     "timestamptype" : {" type" : "string",
            "format": "date-time",
            "description": "ISO8601/RFC3339 formats for date-time ",
     },
     // composite types
     "clause": {
      "type": "object",
      "properties": {
        "name": { "enum": [ "sender", "receiver","time-stamp",
"message-id","in-reply-to","language" ]  } ,
        "value":  {
          "anyOf" : [
            {"$ref" : "#/definitions/integertype" },
            {"$ref" : "#/definitions/numbertype" },
```

```
        {"$ref" : "#/definitions/datetype" },
        {"$ref" : "#/definitions/timestamptype" },
        {"type" : "string"},
      ],
    },
   },
   "required": ["name", "value"]
  },
  //
  "clauses" : { "type" : "array",
     "items" : {"$ref" : "#/definitions/clause"          },
  },

"NumberOfpieces": {
  "type": "object",
  "properties": {
    "name": {"type":"string" },
    "value":  {"$ref" : "#/definitions/integertype" },
  },
  "required": ["name", "value"],
},
"Quantity": {
  "type": "object",
  "properties": {
    "name":{"type": "string" } ,
    "value": {"type" : "object",
            "properties" : {
              "value" :  {"$ref" : "#/definitions/numbertype" },
              "measure" : { "type" : "string" }
            },
            "required" : ["value","measure"],
    },
    "descriptions" : "  ",
```

```
      },
      "required": ["name", "value"],
    },
    "stringValues": {
      "type": "object",
      "properties": {
        "name": {"type": "string" },
        "value":  {
          "anyOf" : [
            {"$ref" : "#/definitions/datetype" },
            {"$ref" : "#/definitions/timestamptype" },
            {"$ref" : "#/definitions/stringtype" },
          ],
        },
        "descriptions": "this object includes not only string value, but
also ISO8601/RFC3339 formats for date, time",
      },
      "required": ["name", "value"],
    },
    "CollectionValues": {
      "type": "object",
      "properties": {
        "name":{"type": "string" } ,
        "value": { "type": "array",
          "items": {
            "$ref" : "#/definitions/nameValuePair",
            },
        },
      },
      "required": ["name", "value"],
    },

   //
```

```
"nameValuePair": {
  "type": "object",
  "properties": {
      "anyOf": {
        "$ref" : "#/definitions/NumberOfpieces",
        "$ref" : "#/definitions/Quantity",
        "$ref" : "#/definitions/stringValues" ,
        "$ref" : "#/definitions/CollectionValues",
       },
    },
},
"ordereditem": {
  "$ref": "#/definitions/nameValuePair"
},
// component
"checker": {
  "type": "object",
  "properties": {
    "name":{ "enum": ["number-of-records", "check-sum"] } ,
    "value":  { "type": "integer" },
   },
  "required": ["name", "value"],
  "minitems":1,
},
"PerformativeMessageBody" : {
   "properties" :{
      "order" : {"type" : "string"},
      "ordereditems" : { "type" : "array",
         "items" : { "type" : "array",
             "items" : {
                "$ref" : "#/definitions/ordereditem"
              },
           },
```

```
            },
          },
          "required" : ["order","ordereditems"],
        },
      },

  "type": "object",
  "properties": {
    // control messages
    "NofifyMessage": {  "type" : "object",
      "properties" :{
       "controlVerb" : {"enum": [ "Notify"]},
       "clauses": {"$ref" : "#/definitions/clauses"},
        "message-content": {"type" : "string" },
       },
        "required": ["controlVerb","clauses","message-content"],
        "additionalProperties": false
    },
    "AskResponseMessage": {  "type" : "object",
      "properties" :{
       "controlVerb" : {"enum": [ "AskResponse"]},
       "clauses": {"$ref" : "#/definitions/clauses"},
       "message-content":  {"type" : "string",
          "pattern" : "^select[\\s]+[\\(]\\*[\\)][\\s]+where.+$" },
       },
        "required": ["controlVerb","clauses","message-content"],
        "additionalProperties": false
    },
    "WarnMessage": {  "type" : "object",
      "properties" :{
       "controlVerb" : {"enum": [ "Warn"]},
       "clauses": {"$ref" : "#/definitions/clauses"},
        "message-content":  {"type" : "string"},
```

```
        },
        "required": ["controlVerb","clauses","message-content"],
        "additionalProperties": false
    },
    // performative messages
    "RequestMessage": { "type" : "object",
        "properties" : {
          "performative" : {"enum": [ "Request"]},
          "clauses":   {"$ref":"#/definitions/clauses"},
        "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
         "checkers" : { "type" : "array",
              "items" : {"$ref" :   "#/definitions/checker" },
          },
        },
        "required": ["performative","clauses","message-content"],
        "additionalProperties": false
    },
    "PromiseMessage": { "type" : "object",
        "properties" : {
          "performative" : {"enum": [ "Promise"]},
          "clauses":  {"$ref":"#/definitions/clauses"},
        "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
         "checkers" : { "type" : "array",
              "items" : {"$ref" :   "#/definitions/checker" },
          },
        },
        "required": ["performative","clauses","message-content"],
        "additionalProperties": false
    },

   "CounterMessage": { "type" : "object",
        "properties" : {
          "performative" : {"enum": [ "Counter"]},
```

```
      "clauses":   {"$ref":"#/definitions/clauses"},
    "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
     "checkers" : { "type" : "array",
          "items" : {"$ref" :   "#/definitions/checker" },
      },
    },
    "required": ["performative","clauses","message-content"],
    "additionalProperties": false
  },
  "AcceptMessage": { "type" : "object",
    "properties" : {
      "performative" : {"enum": [ "Accept"]},
      "clauses":   {"$ref":"#/definitions/clauses"},
    "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
      "checkers" : { "type" : "array",
          "items" : {"$ref" :   "#/definitions/checker" },
      },
    },
    "required": ["performative","clauses","message-content"],
    "additionalProperties": false
  },
  "CancelMessage": { "type" : "object",
    "properties" : {
      "performative" : {"enum": [ "Cancel"]},
      "clauses":   {"$ref":"#/definitions/clauses"},
    "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
      "checkers" : { "type" : "array",
          "items" : {"$ref" :   "#/definitions/checker" },
      },
    },
    "required": ["performative","clauses","message-content"],
    "additionalProperties": false
  },
```

```
"DeclineMessage": { "type" : "object",
  "properties" : {
    "performative" : {"enum": [ "Decline"]},
    "clauses":   {"$ref":"#/definitions/clauses"},
  "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
    "checkers" : { "type" : "array",
        "items" : {"$ref" :   "#/definitions/checker" },
    },
  },
  "required": ["performative","clauses","message-content"],
  "additionalProperties": false
},
"ReportCompletionMessage": { "type" : "object",
  "properties" : {
    "performative" : {"enum": [ "ReportCompletion"]},
    "clauses":   {"$ref":"#/definitions/clauses"},
  "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
    "checkers" : { "type" : "array",
        "items" : {"$ref" :   "#/definitions/checker" },
    },
  },
  "required": ["performative","clauses","message-content"],
  "additionalProperties": false
},
"DeclareCompleMessage": { "type" : "object",
  "properties" : {
    "performative" : {"enum": [ "DeclareComplete"]},
    "clauses":   {"$ref":"#/definitions/clauses"},
  "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
    "checkers" : { "type" : "array",
        "items" : {"$ref" :   "#/definitions/checker" },
    },
  },
```

```
      "required": ["performative","clauses","message-content"],
      "additionalProperties": false
    },
    "DeclineReportMessage": { "type" : "object",
      "properties" : {
        "performative" : {"enum": [ "DeclineReport"]},
        "clauses":   {"$ref":"#/definitions/clauses"},
      "message-content": {"$ref":"#/definitions/PerformativeMessageBody"},
        "checkers" : { "type" : "array",
            "items" : {"$ref" :   "#/definitions/checker" },
        },
      },
      "required": ["performative","clauses","message-content"],
      "additionalProperties": false
    },
  },
}
```

Annex B
(informative)

Management Layer

B.1 Layer of Production Management System

Production management system consists of subsystems. The subsystem of the production management system assumed in this document is shown in Figure B.1. And, In addition, the subsystems of this document can correspond to functional hierarchies from level 0 to level 4 defined by IEC 62264.

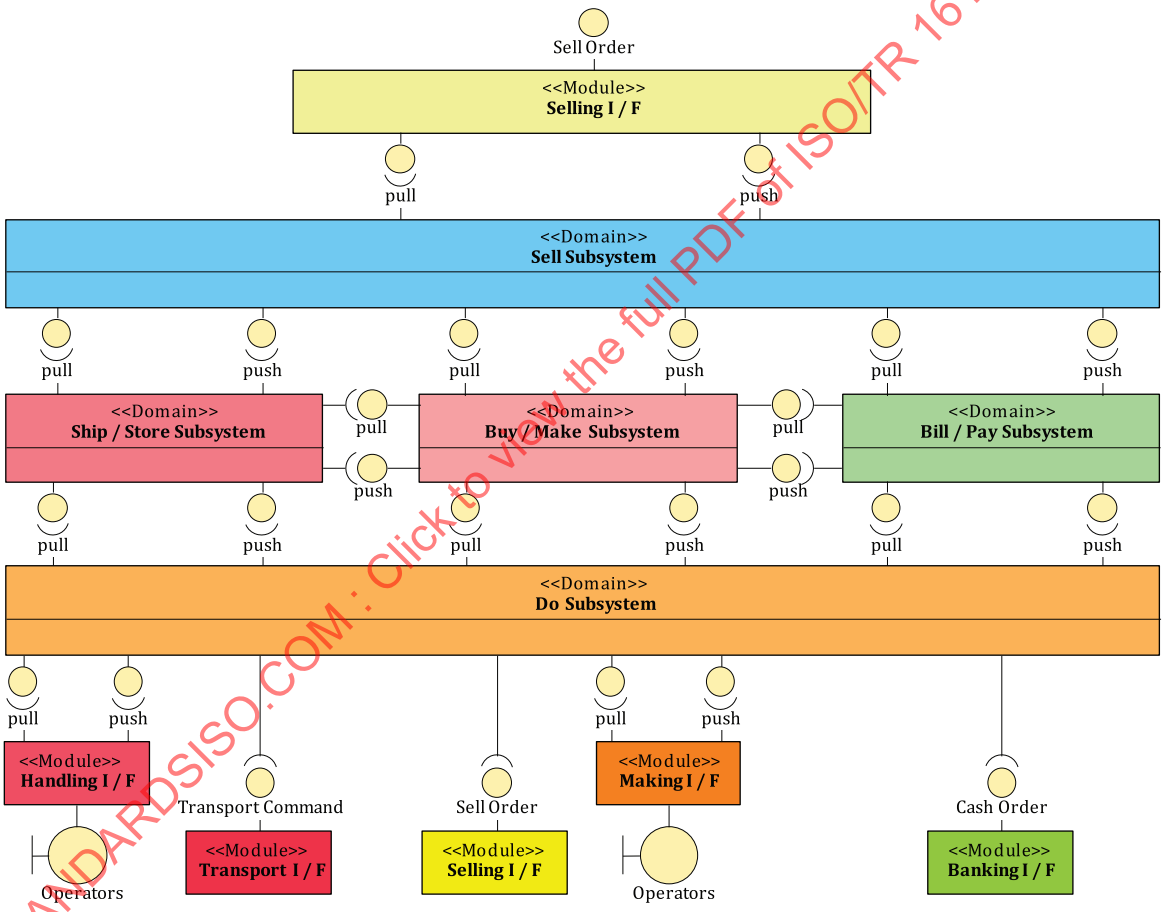

Figure B.1 — Overview of Production Management System

B.2 Subsystems

a) Sell subsystem: The Sell subsystem deals with the life cycle of orders received from confirmation of order to completion of order (request or deposit). This subsystem corresponds to level 4 of IEC 62264 functional hierarchies. The order is either purchase (Buy) or production (Make) depending on the item.

b) Buy/Make Subsystem: The Buy/Make Subsystem plans as a supplier to a purchase or production order (Buy/Make) issued at the Sell subsystem. This subsystem corresponds to level 4 of IEC 62264 functional hierarchies.

c) Ship/Store Subsystem: The Ship/Store Subsystem plans logistics work such as acceptance and delivery of parts created or purchased in the Buy/Make subsystem. This subsystem corresponds to level 4 of IEC 62264 functional hierarchies.

d) Claim/Pay Subsystem: The Claim/Pay subsystem makes a plan of payment of the price billing and purchasing, purchasing to the order issued by the Sell subsystem. This subsystem corresponds to level 4 of IEC 62264 functional hierarchies.

e) Do Subsystem: The Do Subsystem concretely implements orders of logistics, manufacturing and purchasing orders, orders for deposits and orders planned by the upper subsystem, controls logistics or manufacturing equipment, and records the results.

B.3 Examples of the order in Request Message

The order's value depends on subsystem. Examples of frequently used order's values are listed in Table B.1.

Table B.1 — Frequently used order's values

C-subsystem	order	Description	P-subsystem	who	what	how many	how much	when by	where from	where to
Selling I/F	Sell	Sell goods	Sell	customer	goods	pieces/ quantity	payment	due date	—	destination
Sell	Ship	Ship goods	Ship/Store	sales	goods	pieces/ quantity	—	shipping date	shipping source	destination
Sell	Buy	Buy goods	Buy/Make	sales	goods	pieces/ quantity	payment	arrival date	purchaser	warehouse
Sell	Make	Make goods	Buy/Make	sales	goods	pieces/ quantity	—	completion date	—	warehouse
Sell	Store	Receive goods	Ship/Store	sales	goods	pieces/ quantity	—	arrival date	shipping source	warehouse
Sell	Claim	Payment will be accounts receivable	Claim/Pay	sales	account	—	payment	accrual date	—	billing destination
Buy/Make	Buy	Buy parts	Buy/Make	factory	items	pieces/ quantity	—	due date	purchaser	factory
Buy/Make	Make	Make parts	Buy/Make	factory	items	pieces/ quantity	—	completion date	factory	Factory
Buy/Make	Store	Receive parts	Ship/Store	Factory	items	pieces/ quantity	—	arrival date	shipping source	warehouse
Buy/Make	Ship	Issue the product	Ship/Store	factory[2]	items	pieces/ quantity	—	issue date	factory	warehouse
Buy/Make	Store	Receive the product	Ship/Store	factory	items	pieces/ quantity	—	receipt date	factory	warehouse
Buy/Make	Pay	Payment will be accounts receivable	Claim/Pay	purchase	account	—	payment	accrual date	—	payee
Buy/Make	Do	Do make	Do(Make)	factory	—	pieces/ quantity	—	—	—	—
Ship/Store	Do	Do shipping	Do(Ship)	Warehouse	—	—	—	—	warehouse	transport company

Table B.1 *(continued)*

C-subsystem	order	Description	P-subsystem	who	what	how many	how much	when by	where from	where to
Ship/Store	Do	Do receive	Do(Store)	warehouse	—	—	—	—	transport company	warehouse
Claim/Pay	Do	Claim money	Do(Claim)	sales	account	—	payment	billing date	—	billing destination
Claim/Pay	Do	Pay the amount to be invoiced	Do(Pay)	purchase	account	—	payment	pay day	—	payee

B.4 Collaboration of C-subsystem, P-subsystem and service provider

The interaction function is described in Collaboration Diagram of UML 2.4. An interaction between MSU is expressed using Collaboration Use of UML that is an occurrence Collaboration. The interaction between the MSUs is shown in Figure B.2. Connection between Subsystems is explained using this. The MSU that interacts has an occurrence of C - P dialogue. In Figure B.2, the MSU of the A Subsystem and the MSU of the B Subsystem show the interaction via the occurrence of the C-P dialogue. The rectangle in Figure B.2 shows the MSU, and the ellipse shows the function. The text adjacent to the line connecting the MSU and the function shows the Lifeline of the C-P dialogue function. In the dialogue between A and B, Figure B.2 shows that the role of C-subsystem of C-P dialogue is assigned to A and the role of P-subsystem of C-P dialogue is assigned to B.

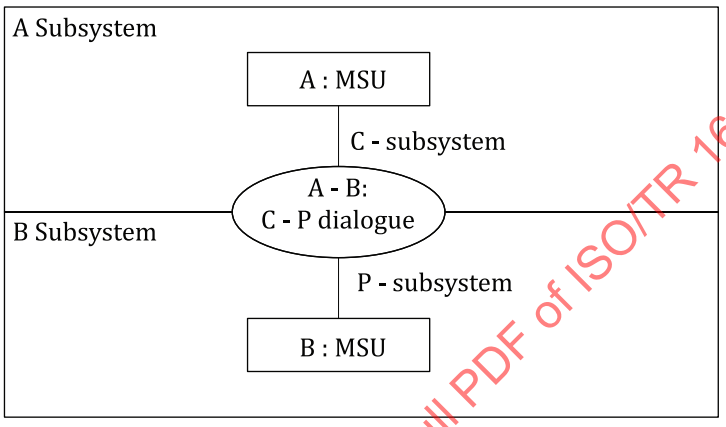

Figure B.2 — Instance of C-P dialogue between A Subsystem and B Subsystem

Figure B.3 shows that the Sell Subsystem interacts with the MSU of the Store Subsystem,

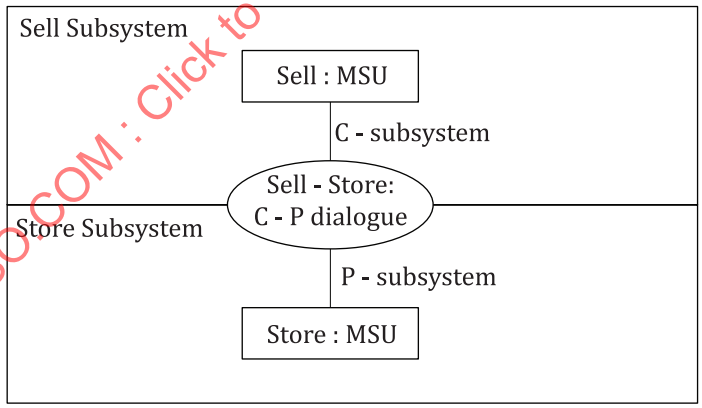

Figure B.3 — Instance of C-P dialogue between Sell Subsystem and Store Subsystem

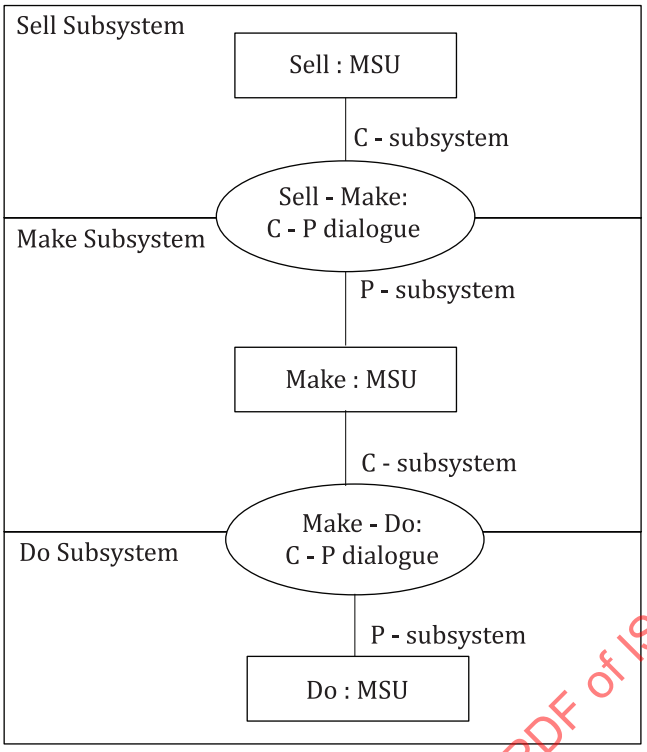

Figure B.4 — Instances of C-P dialogue between Sell Subsystem and Make Subsystem and between Make subsystem and Do Subsystem

Figure B.4 shows that Make Subsystem interacts with Do subsystem, and that one MSU has two roles (P-subsystem and C-subsystem), Sell Subsystem is the role of P-subsystem, Do subsystem is C-subsystem role.

B.5 Sequence diagram of C-P dialogue base

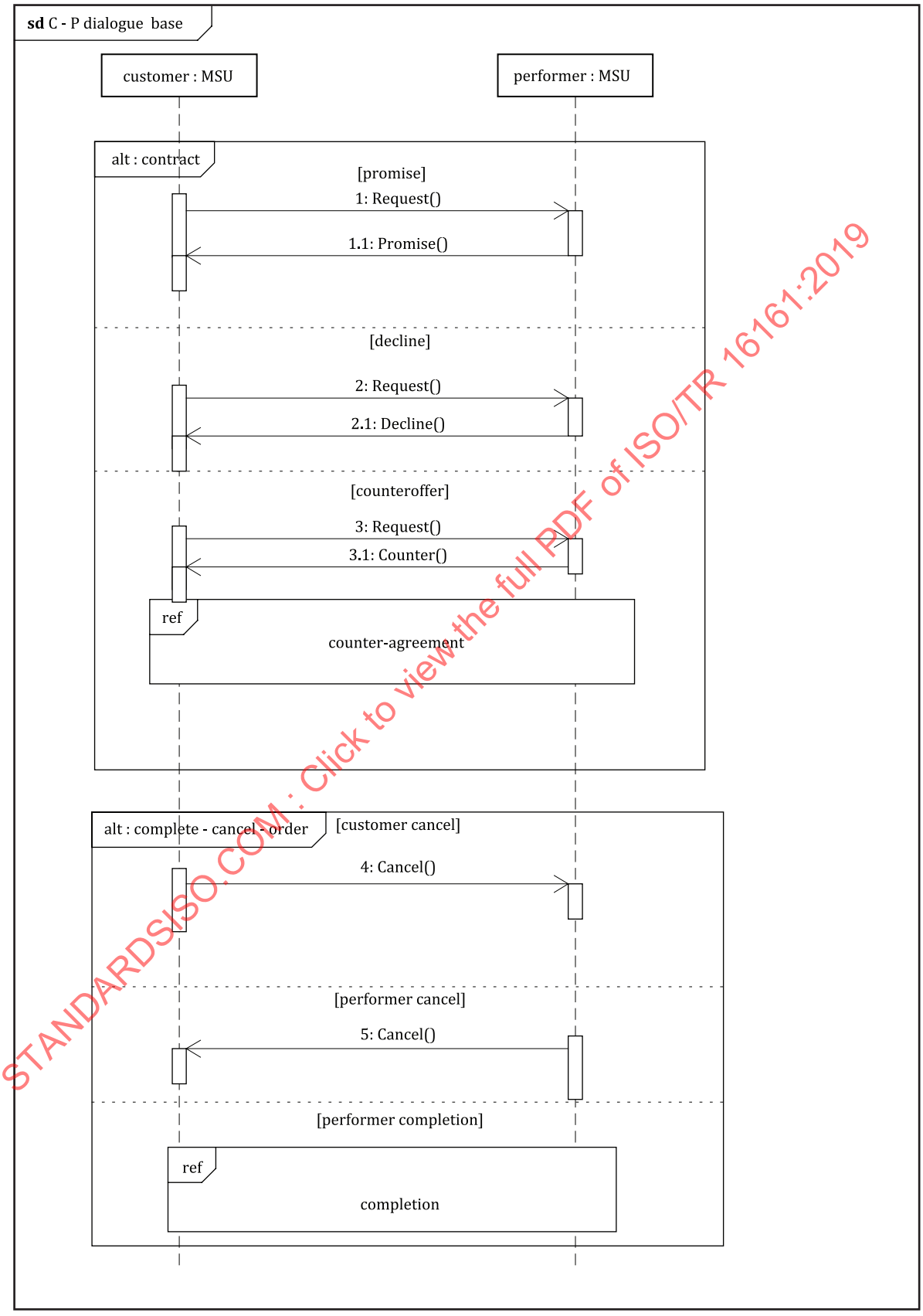

Figure B.5 — Sequence diagram of C-P dialogue base

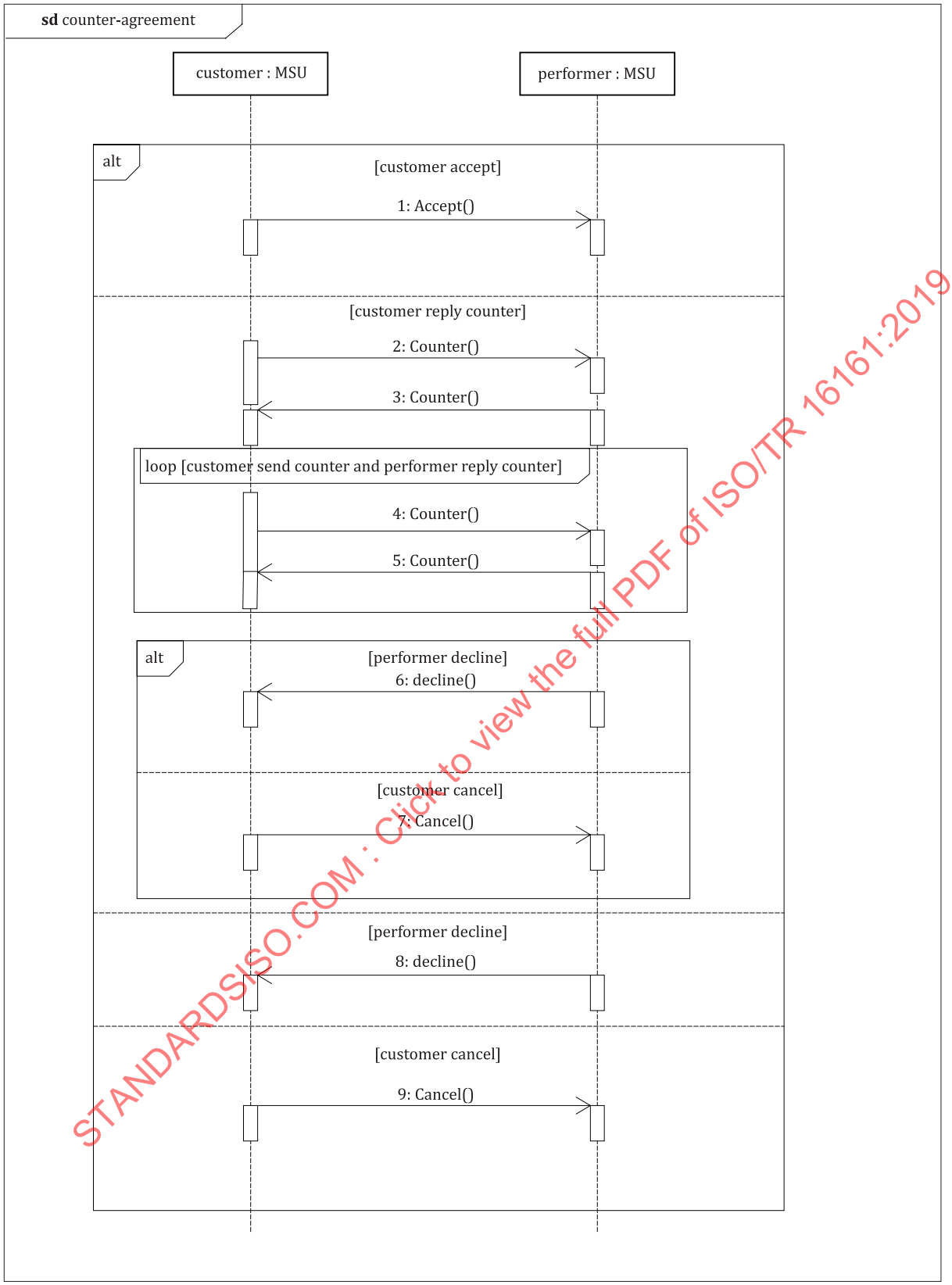

Figure B.6 — Sequence diagram of counter-agreement

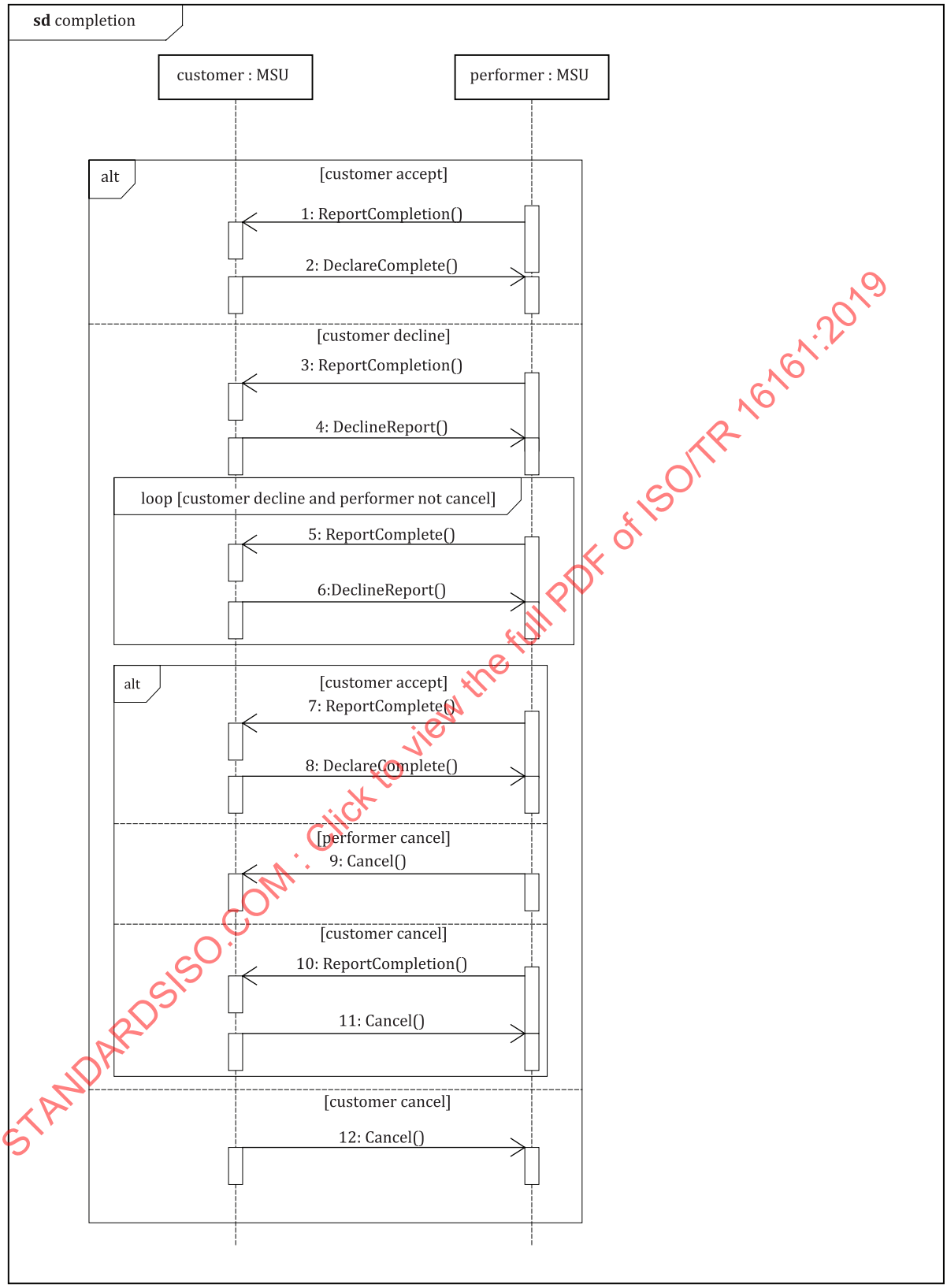

Figure B.7 — Sequence diagram of completion

Annex C
(informative)

Customer-performer model

The dialogue between customer and performer usually transits to states 1, 2, 3, 4, and 5 shown in the transition diagram of the dialogue of Figure 1. In the following description, the state shown in the transition diagram of the dialogue in Figure 1 is used. Figure C.1 shows that the flow of information between ordinary customer and performer represented by Figure 1 corresponds to the state of customer and performer.

The performer can decline the customer's order in state 2 and cancel in state 3. After declining or withdrawing, customer-performer dialogue is over.

The customer can cancel the order issued by itself in state 2 ', state 3, state 4. After cancellation, the customer-performer dialogue is over.

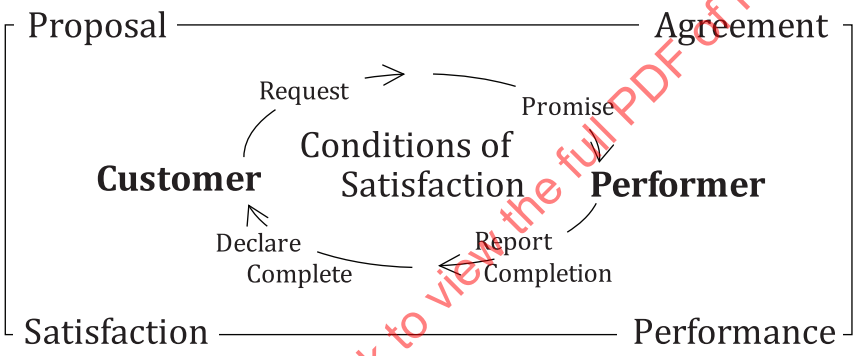

Figure C.1 — Customer-performer model

— The performer can negotiate the order contents of the customer.

— The performer proposes a counteroffer to the order contents to the customer and transits to State 2'.

— The customer selects one of the following actions.

— Accept counteroffer of the performer transit from state 2 'to state 3.

— Reject the counteroffer of the performer and end the dialogue.

— Send a counteroffer to the counteroffer of the performer to the performer and make a transition to State 2.

— When the customer returns a counteroffer to the performer, the performer selects one of the following actions.

— Decline and end the dialogue.

— Accept, return promise message to the customer and transition to State 3.

— Respond to the customer's countermeasure to the customer and return to state 2.