INTERNATIONAL STANDARD

ISO/IEC 23006-5

First edition
2013-04-01

Information technology — Multimedia service platform technologies —

Part 5: Service aggregation

Technologies de l'information — Technologies de la plate-forme de services multimédia —

Partie 5: Aggrégation de service

Reference number
ISO/IEC 23006-5:2013(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23006-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23006 consists of the following parts, under the general title *Information technology — Multimedia service platform technologies*:

— *Part 1: Architecture*

— *Part 2: MPEG extensible middleware (MXM) API*

— *Part 3: Conformance and reference software*

— *Part 4: Elementary services*

— *Part 5: Service aggregation*

Introduction

ISO/IEC 23006 is a suite of standards that has been developed for the purpose of enabling the easy design and implementation of media-handling value chains whose devices interoperate because they are all based on the same set of technologies accessible from the middleware.

ISO/IEC 23006 was referred to as MPEG Extensible Middleware (MXM) in its first edition, and it specified an architecture (Part 1), an API (Part 2), a reference software (Part 3) and a set of protocols to which MXM Devices had to adhere (Part 4).

ISO/IEC 23006 is referred to as Multimedia Service Platform Technologies (MSPT) in its second edition, and it conserves the architecture and design philosophy of the first edition, but stresses the Service Oriented Architecture character. It also specifies how to combine elementary services into aggregated services (Part 5).

This second edition has been specified to address the demand of service specification for an advanced IPTV terminal (AIT). The ISO/IEC 23006 suite of standards also aims at leveraging on advanced technologies to bring into IPTV services the buoyancy of new exciting initiatives – sometimes assembling millions of users in a fortnight – that pop up almost every day with new features such as open APIs and the possibility for third parties to provide applications to those APIs.

This enables the development of a global market of:

— MSPT applications that can run on MSPT devices, such as Advanced IPTV Terminals (AITs), thanks to the existence of a standard MSPT application API;

— MSPT devices executing MSPT applications thanks to the existence of a standard MSPT architecture;

— MSPT engines thanks to the existence of standard MSPT architecture and standard APIs;

— Innovative business models because of the ease to design and implement media-handling value chains whose devices interoperate because they are all based on the same set of technologies, especially MPEG technologies.

Information technology — Multimedia service platform technologies —

Part 5:
Service aggregation

1 Scope

This part of ISO/IEC 23006 specifies the technology enabling the combination of Elementary Services (ESs) to build Aggregated Services (ASs). It is worth noting that it does not impose the use of any set of technologies but provides a methodology which defines the basic steps for the definition of new ASs and gives some representative examples to help in the understanding of the methodology.

It also describes the mechanism for registration of ASs and new ESs not present in ISO/IEC 23006-4. The services registered could be then used in the same way as the ones defined in ISO/IEC 23006-4 for further usage, for instance, for the development of new Aggregated Services.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23006-4, *Information technology — Multimedia service platform technologies — Part 4: Elementary services*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

3.1.1
aggregated service
service resulting from the combination of Elementary Services or other External or Aggregated Service

Note 1 to entry: See further description in Clause 6.

3.1.2
elementary service
basic unit of service

3.1.3
content
digital item and its component elements, namely resources, identifiers, descriptions

EXAMPLE: Resources: media, scripts, executable.

EXAMPLE: Descriptions: metadata.

3.1.4
contract
representation of agreements formed for the transaction of MPEG-21 Digital Items or services related to the MPEG-21 Framework

3.1.5
device
hardware/software or simply software apparatus that enables a **user** to play a role in multimedia value chains

3.1.6
event
performance of a specified set of functions or operations

3.1.7
entity
one of the following elements in the multimedia value chain: **content**, **contract**, **device**, **event**, **license**, **service**, and **user**

3.1.8
governance
ability to control, direct or oversee the behavior of each **entity** or operation in an multimedia value chain

3.1.9
license
collection of authorisations, conditions and payment terms granted by a **user** to other **user**s

3.1.10
protocol
set of rules and data format used by two devices to communicate

3.1.11
resource
individually identifiable asset or a sequence of assets

EXAMPLE: A video or audio clip, a 3D synthetic scene, an image, a textual asset, a 2D LASeR scene, a web page, a single program or a full 24 hour programming of a TV broadcast, a script or executable etc.

3.1.12
right
ability of a **user** to perform an operation in the multimedia **value chain**

3.1.13
role
ability of a **user** to perform a set of operations in the multimedia **value chain**

3.1.14
service
set of operations performed by a **user** on behalf of other **user**s

3.1.15
service definition
specification of a service in terms of interfaces, protocols as well as syntax and semantics of the protocol data formats

3.1.16
service instance declaration
description of a particular implementation of a service in terms of provider, connection end-points, and usage conditions

3.1.17
service provider
user offering **service**s to other **user**s

3.1.18
tag
free text descriptive information attached to an **entity**

3.1.19
user
any participant in multimedia **value chains**

3.1.20
value chain
collection of **users**, including creators, end users and service providers, that conform to this part of ISO/IEC 23006

3.2 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply:

API Application Programming Interface

AS Aggregated Service

BPMN Business Process Model and Notation

DI Digital Item

DIDL Digital Item Declaration Language

ES Elementary Service

IPTV Internet Protocol Television

MPEG Moving Picture Experts Group

MSPT Multimedia Service Platform Technologies

OMG Object Management Group

RA Registration Authority

SP Service Provider

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

4 Namespaces and conventions

4.1 Namespaces

The value of the namespace URI [3] for Schema definitions of the BPMN 2.0 extension for message flow references in this part of ISO/IEC 23006 is:

```
urn:mpeg:mpegM:schema:06-bpmn-ext-st-NS:2012
```

4.2 Namespace convention

For clarity, throughout this part of ISO/IEC 23006, consistent namespace prefixes are used.

"`xml:`" and "`xmlns:`" are normative prefixes defined in [2]. The prefix "`xml:`" is by definition bound to "`http://www.w3.org/XML/1998/namespace`". The prefix "`xmlns:`" is used only for namespace bindings and is not itself bound to any namespace name.

"`xsi:`" prefix is not normative. It is a naming convention in this document to refer to an element of the `http://www.w3.org/2001/XMLSchema-instance` namespace. All other prefixes used in either the text or examples of this specification are not normative, e.g., "`mpegmb:`", "`bpmn:`".

In particular, most of the informative examples in this specification are provided as XML fragments without the normally required XML document declaration and, thus, miss a correct namespace binding context declaration.

Unless specified otherwise, all unqualified descriptions fragments assume the default namespace "`urn:mpeg:mpegM:schema:02-service-NS:2012`".

In these descriptions fragments the different prefixes are bound to the namespaces as given in **Table 1**. The schema locations of the namespaces in **Table 1** are only an informative indication as schema locations may change over time.

Table 1 — Mapping of prefixes to namespaces used in examples and text.

Prefix	Corresponding namespace	Schema location
xsd	http://www.w3.org/2001/XMLSchema	
xsi	http://www.w3.org/2001/XMLSchema-instance	
dia	urn:mpeg:mpeg21:2003:01-DIA-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/dia-2nd/UED-2nd.xsd
bpmn	http://www.omg.org/spec/BPMN/20100524/MODEL	http://www.omg.org/spec/BPMN/20100501/BPMN20.xsd
bpmnext2	urn:mpeg:mpegM:schema:06-bpmn-ext-st-NS:2012	
mpeg7	urn:mpeg:mpeg7:schema:2004	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-7_schema_files/mpeg7-v2.xsd
mpegmb	urn:mpeg:mpegM:schema:01-base-NS:2012	
mpegm	urn:mpeg:mpegM:schema:02-service-NS:2012	

Unlike the informative descriptions examples, the normative specification of the syntax of tools in XML Schema follows the namespace binding context defined in the relevant schema declaration.

4.3 Conventions

Fixed-width font is used to indicate literal machine-readable character sequences.

The names of XML Schema attributes appear in fixed-width font in mixed case with an initial lower case letter. The names of XML Schema elements and types appear in fixed-width font in mixed case with an initial upper case letter.

XML Schema definitions are represented with orange amber background color and a double line black border.

XML examples are represented with gray background color and a single line black border.

XML Infosets of MPEG-7 Classification Schemes, XML declarations of Service Types and BPMN 2.0 XML notations are represented with gray background color and a double line black border.

4.4 MPEG-M Elementary and Aggregated Services Registration Authority

The MPEG-M Services RA is the organization that supports the registration and request of Elementary and Aggregated Services.

Registration of Elementary and Aggregated Services shall follow the guidelines included in this part of ISO/IEC 23006.

5 Overview

The scope of the MSPT is to support the service providers' drive to deploy innovative multimedia services identifying a set of ESs and defining the corresponding set of protocols and APIs to enable any user in an MSPT value chain to access those services in an interoperable manner. This part of ISO/IEC 23006 specifies a methodology to describe the aggregation of Services in MSPT. The aim of this methodology is to help both users and Service Providers (SPs) in the definition of ASs that fit their needs and provide the means for a common understanding between the different parties involved in the service.

ISO/IEC 23006-4 defines a set of ESs that shall be used in MSPT value chains. However, this International Standard allows for external organizations to define additional Services. An Aggregated Service can be composed of one or more Services. These composing Services can be either Elementary Services (cf. ISO/IEC 23006-4:2012), Aggregated Services themselves or external Elementary Services not present in ISO/IEC 23006-4:2012 but described following the same design principles as ESs, as defined in 5.1.3 of ISO/IEC 23006-4:2012.

The underlying idea is that, using ISO/IEC 23006-5 Aggregated Services definition methodology, a user may define his service requirements for a specific MSPT value chain as an aggregated service and give them to a service provider (SP) in order to ask this SP for an implementation of the aggregated service provided.

The rest of the document is organized as follows:

- Clause 6 specifies how Aggregated Services should be defined and registered.
- Clause 7 defines process modeling representations that can be used for the definition of AS.

6 Aggregated Service definition

6.1 Introduction

This clause describes the details for the complete definition of an Aggregated Service, including the methodology to be used for its definition, the MPEG-M Registration Authority for new Elementary and Aggregated Services and some examples to clarify the usage of the methodology.

6.2 Methodology

This subclause specifies the methodology for the description of an Aggregated Service. The definition of Aggregated Services takes some elements from the General Service Definition for Elementary Services as specified in 5.1.3 of ISO/IEC 23006-4:2012, but it also needs some specific steps. They are described in detail as follows.

Aggregated Services may define their own messages in addition to those that are used by the composing Elementary Services. The messages of Aggregated Services must be specified following the rules provided in 5.1.3 of ISO/IEC 23006-4:2012.

The methodology for the definition of new aggregated services comprises the following steps:

1) Provision of a narrative description of the actions that the Service performs, that is, a general textual description of the use case or scenario to be described as AS.

2) Identification of Elementary and Aggregated Services that are part of the AS to be defined. These ESs and ASs could be:

   2.1) ESs described in ISO/IEC 23006-4:2012 and/or ASs described in ISO/IEC 23006-5.

   2.2) MPEG-M compliant ESs and/or ASs registered in the Registration Authority (RA) described in 6.3 of this part of ISO/IEC 23006.

   2.3) External ESs specifically defined for the AS being defined. In this case, the messages must be specified following the rules provided in 5.1.3 of ISO/IEC 23006-4:2012. This new ESs can be registered in the RA, described in 6.3, for further use.

3) Provision of a textual description of the AS service workflow describing interaction between Client and Service Provider making use of the following table:

Steps	Service invoked	Client + Message	Service Provider + Message	Description
Order of the operations involved.	Elementary (ES), External (EES) or Aggregated service (AS) involved. ES, EES or AS is added after service's name.	Who is taking the Client's role and which message is sent. Message is only indicated when Client is making a request.	Who is taking the Service Provider role and which message is sent. Message is only indicated when Service Provider is giving a response.	Textual description of what is done in this step.

4) Provision of a formal description of the service workflow of the resulting AS. It should describe both protocol and service by including the following elements:

   - Graphical service workflow diagram (see Clause 7 for guidelines).

- [Optional] Service workflow XML serialization (see Clause 7 for guidelines).

5) Optional registration of the resulting AS in the RA specified in 6.3. The RA will syntactically validate the registered AS.

6.3 MPEG-M Elementary and Aggregated Services Registration Authority

This subclause targets on the definition of the required information for the registration process of Elementary and Aggregated Services. The process of registration is recommended for those ES that want to be compliant with ISO/IEC 23006-4:2012 and for those AS that want to be compliant with ISO/IEC 23006-5. Furthermore, the registration will allow those Services to be used by other RA users.

The information required for the registration of an ES is the one described in 5.1.3 of ISO/IEC 23006-4:2012, which is protocol, service, formal representation, including the service workflow diagram, and its corresponding XML serialization.

The information required for the registration of an AS is the one described in 6.2.

This part of ISO/IEC 23006 supports the scenario where there is a global authority for storing ESs and ASs information. This authority will be established for permitting the obtaining of this information to all MPEG-M compliant applications. With this, applications will be able to access and use the registered ESs and ASs. The list of ISO registration authorities can be found at http://www.iso.org/iso/maintenance_agencies. The detailed process of registration can be found in Annex A.

6.4 Examples

6.4.1 Introduction

The example described in this subclause illustrates how the methodology has to be applied to the definition of a new AS formed by existing ES.

The AS defined next represents a web-based application for selling and purchasing digital objects. It can be divided into two differentiated use cases or scenarios, the seller use case and the buyer use case. Both are described in this subclause for completeness. It is worth noting that the described web-based application can be executed on different kinds of devices (PC, smartphone, etc.) but the resulting AS is independent from the device used by the client to interact with the service provider.

6.4.2 Application of the methodology to the seller use case

Step 1. Narrative description

In this use case, a content creator wants to register some multimedia content she has created (music score, video, photo or a combination of them) for putting it on sale. To do so, she connects to a web-based application which is the front-end to a MPEG-M compliant system which provides her with the functionality requested, that is, content registration, content storage, licensing capabilities and event reporting, among others.

Step 2. Identification of ESs and ASs present in the AS

The ESs to be used are: Authenticate User, Identify Content, Create Content, Store Content, Create License, Store License and Store Event. All of them are defined in ISO/IEC 23006-4:2012.

Step 3. Workflow textual description

The service workflow associated to the Register and Sell Content aggregated service (RAS) is as follows:

Steps	Service invoked	Client + Message	Service Provider + Message	Description
1.	Authenticate User (ES)	RAS Client `AuthenticateUserRequest`	Authenticate User SP	*RAS Client* tries to authenticate in the system.
2.	Authenticate User (ES)	RAS Client	Authenticate User SP `AuthenticateUserResponse`	*Authenticate User SP* tries to authenticate RAS Client with the information provided. If response is negative, AS stops here.
3.	Create Content (ES)	RAS Client `CreateContentRequest`	Create Content SP	*RAS Client* provides the required information to create some content. This information may include content metadata, resources and licenses (offers at this step).
4.	Create Content (ES)	RAS Client	Create Content SP `CreateContentResponse`	*Create Content SP* creates the content with the information sent from client. If this operation is not successful, AS stops here.
5.	Identify Content (ES)	RAS Client `IdentifyContentRequest`	Identify Content SP	*RAS Client* requests an identifier for the newly created content.
6.	Identify Content (ES)	RAS Client	Identify Content SP `IdentifyContentResponse`	*Identify Content SP* generates an identifier for the content. If this operation is not successful, AS stops here.
7.	Store Content (ES)	RAS Client `StoreContentRequest`	Store Content SP	*RAS Client* requests content to be stored.
8.	Store Content (ES)	RAS Client	Store Content SP `StoreContentResponse`	*Store Content SP* stores the identified content sent by the client. If this operation is not successful, AS stops here.
9.	Store Event (ES)	RAS Client `StoreEventRequest`	Store Event SP	*RAS Client* requests event to be stored. (This step is done in parallel with step 11).
10.	Store Event (ES)	RAS Client	Store Event SP `StoreEventResponse`	*Store Event SP* stores an event informing of the content stored.
11.	Create License (ES)	RAS Client `CreateLicenseRequest`	Create License SP	*RAS Client* provides the required information to create some licenses over content created in previous steps.

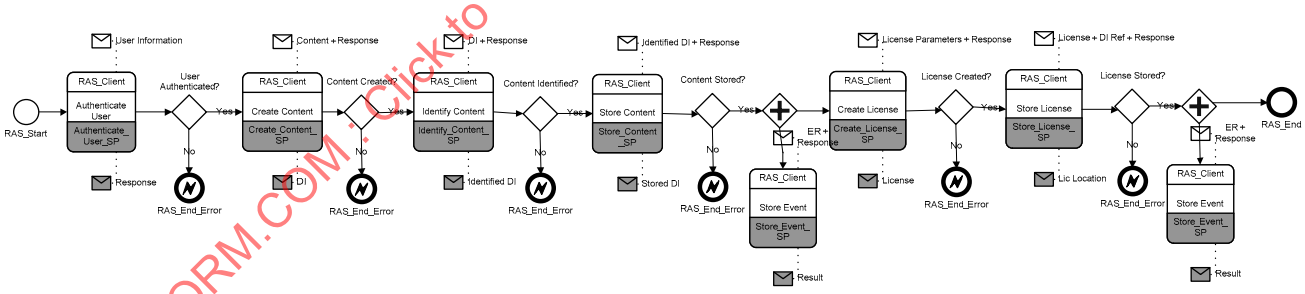
Steps	Service invoked	Client + Message	Service Provider + Message	Description
12.	Create License (ES)	RAS Client	Create License SP `CreateLicenseResponse`	*Create License SP* creates the license(s) with the information sent from client. If this operation is not successful, AS stops here.
13.	Store License (ES)	RAS Client `StoreLicenseRequest`	Store License SP	*RAS Client* requests license storage.
14.	Store License (ES)	RAS Client	Store License SP `StoreLicenseResponse`	*Store License SP* stores the licenses sent by RAS Client. If this operation is not successful, AS stops here.
15.	Store Event (ES)	RAS Client `StoreEventRequest`	Store Event SP	*RAS Client* requests event to be stored.
16.	Store Event (ES)	RAS Client	Store Event SP `StoreEventResponse`	*Store Event SP* stores an event informing of the licenses stored.

Step 4. Formal description of the service workflow

The methodology does not define which representation has to be used for the formal description of the service workflow, just that a diagram and an optional XML serialization should be provided. In this example, BPMN 2.0 is used for the formal description of the proposed AS (see Clause 7 for details).

This description is as follows:

BPMN 2.0 Diagram



BPMN 2.0 XML serialization

```
<bpmn:definitions id="RAS_Choreography" name="RAS_Choreography" targetNamespace="ras"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:semantic="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:mpegm="urn:mpeg:mpegM:service:01-service-NS:2012">
    <bpmn:choreography id="RAS_Service" name="Register And Sell content">
        <bpmn:participant name="RAS_Client" id="ras_client"/>
        <bpmn:participant name="Authenticate_User_SP" id="authenticate_user_sp"/>
        <bpmn:participant name="Create_Content_SP" id="create_content_sp"/>
        <bpmn:participant name="Identify_Content_SP" id="identify_content_sp"/>
```

```
        <bpmn:participant name="Store Content SP" id="store content sp"/>
        <bpmn:participant name="Create_License_SP" id="create_license_sp"/>
        <bpmn:participant name="Store_Event_SP" id="store_event_sp"/>
        <bpmn:participant name="Store License SP" id="store license sp"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="authenticate_user_sp"
id="authenticate_user_req_message_flow" messageRef="mpegm:authenticate_user_request"/>
        <bpmn:messageFlow sourceRef="authenticate user sp" targetRef="ras client"
id="authenticate_user_rsp_message_flow" messageRef="mpegm:authenticate_user_response"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="create_content_sp"
id="create_content_req_message_flow" messageRef="mpegm:create_content_request"/>
        <bpmn:messageFlow sourceRef="create_content_sp" targetRef="ras_client"
id="create_content_rsp_message_flow" messageRef="mpegm:create_content_responset"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="identify_content_sp"
id="identify_content_req_message_flow" messageRef="mpegm:identify_content_request"/>
        <bpmn:messageFlow sourceRef="identify_content_sp" targetRef="ras_client"
id="identify_content_rsp_message_flow" messageRef="mpegm:identify_content_response"/>
        <bpmn:messageFlow sourceRef="ras client" targetRef="store content sp"
id="store_content_req_message_flow" messageRef="mpegm:store_content_request"/>
        <bpmn:messageFlow sourceRef="store_content_sp" targetRef="ras_client"
id="store content rsp message flow" messageRef="mpegm:store content response"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="create_license_sp"
id="create_license_req_message_flow" messageRef="mpegm:create_license_request"/>
        <bpmn:messageFlow sourceRef="create license sp" targetRef="ras client"
id="create_license_rsp_message_flow" messageRef="mpegm:create_license_response"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="store_event_sp"
id="store event req message flow" messageRef="mpegm:store event request"/>
        <bpmn:messageFlow sourceRef="store_event_sp" targetRef="ras_client"
id="store_event_rsp_message_flow" messageRef="mpegm:store_event_response"/>
        <bpmn:messageFlow sourceRef="ras client" targetRef="store license sp"
id="store_license_req_message_flow" messageRef="mpegm:store_license_request"/>
        <bpmn:messageFlow sourceRef="store_license_sp" targetRef="ras_client"
id="store license rsp message flow" messageRef="mpegm:store license response"/>
        <bpmn:messageFlow sourceRef="ras_client" targetRef="store_event_sp"
id="store_event_req_message_flow2" messageRef="mpegm:store_event_request"/>
        <bpmn:messageFlow sourceRef="store_event_sp" targetRef="ras client"
id="store_event_rsp_message_flow2" messageRef="mpegm:store_event_response"/>
        <bpmn:startEvent id="RAS_Start" isInterrupting="true" name="RAS_Start"
parallelMultiple="false">
            <bpmn:outgoing>authenticate_user_ES</bpmn:outgoing>
        </bpmn:startEvent>
        <bpmn:choreographyTask initiatingParticipantRef="ras_client" name="Authenticate User"
id="authenticate_user_task">
            <bpmn:outgoing>is_user_authenticated</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>authenticate_user_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>authenticate_user_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>authenticate_user_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="User Authenticated?"
id="gw_user_authenticate">
            <bpmn:incoming>is user authenticated</bpmn:incoming>
            <bpmn:outgoing>user_authenticate_yes</bpmn:outgoing>
            <bpmn:outgoing>user_authenticate_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS_End_Error" id="end_not_authenticated">
            <bpmn:incoming>user_authenticate_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
```

```
        <bpmn:choreographyTask initiatingParticipantRef="ras client" name="Create Content"
id="create_content_task">
            <bpmn:incoming>user_authenticate_yes</bpmn:incoming>
            <bpmn:outgoing>is content created</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>create_content_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>create content req message flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>create_content_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="Content Created?"
id="gw_content_created">
            <bpmn:incoming>is_content_created</bpmn:incoming>
            <bpmn:outgoing>content_created_yes</bpmn:outgoing>
            <bpmn:outgoing>content_created_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS_End_Error" id="end_not_created">
            <bpmn:incoming>content created no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="ras client" name="Identify Content"
id="identify_content_task">
            <bpmn:incoming>content_created_yes</bpmn:incoming>
            <bpmn:outgoing>is content identified</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>identify_content_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>identify content req message flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>identify_content_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="Content Identified?"
id="gw_content_identified">
            <bpmn:incoming>is_content_identified</bpmn:incoming>
            <bpmn:outgoing>content identified yes</bpmn:outgoing>
            <bpmn:outgoing>content_identified_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS End Error" id="end not identified">
            <bpmn:incoming>content_identified_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="ras_client" name="Store Content"
id="store_content_task">
            <bpmn:incoming>content_identified_yes</bpmn:incoming>
            <bpmn:outgoing>is_content_stored</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>store_content_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_content_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store_content_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="Content Stored?"
id="gw_content_store">
            <bpmn:incoming>is_content_stored</bpmn:incoming>
            <bpmn:outgoing>content stored yes</bpmn:outgoing>
            <bpmn:outgoing>content_stored_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS End Error" id="end not stored">
            <bpmn:incoming>content_stored_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:parallelGateway gatewayDirection="Unspecified" name="Store Event and Store Content"
id="par_gw_store_event_content">
```

```
            <bpmn:incoming>content stored yes</bpmn:incoming>
            <bpmn:outgoing>event_and_content_stored</bpmn:outgoing>
            <bpmn:outgoing>store_event_content</bpmn:outgoing>
        </bpmn:parallelGateway>
        <bpmn:choreographyTask initiatingParticipantRef="ras_client" name="Create License"
id="create_license_task">
            <bpmn:incoming>event and content stored</bpmn:incoming>
            <bpmn:outgoing>is_license_created</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>create_license_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>create_license_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>create_license_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:choreographyTask initiatingParticipantRef="ras_client" name="Store Event"
id="store_event_content_task">
            <bpmn:incoming>store_event_content</bpmn:incoming>
            <bpmn:participantRef>ras client</bpmn:participantRef>
            <bpmn:participantRef>store_event_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_event_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store event rsp message flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="License Created?"
id="gw license created">
            <bpmn:incoming>is_license_created</bpmn:incoming>
            <bpmn:outgoing>license_created_yes</bpmn:outgoing>
            <bpmn:outgoing>license created no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS_End_Error" id="end_no_license">
            <bpmn:incoming>license created no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="ras client" name="Store License"
id="store_license_task">
            <bpmn:incoming>license_created_yes</bpmn:incoming>
            <bpmn:outgoing>event and license stored</bpmn:outgoing>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
            <bpmn:participantRef>store_license_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store license req message flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store_license_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="License Stored?"
id="gw_license_stored">
            <bpmn:incoming>event_and_license_stored</bpmn:incoming>
            <bpmn:outgoing>license_stored_yes</bpmn:outgoing>
            <bpmn:outgoing>license_stored_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="RAS_End_Error" id="end_no_stored_license">
            <bpmn:incoming>license stored no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:parallelGateway gatewayDirection="Unspecified" name="Store Event and Store License"
id="par_gw_store_event_license">
            <bpmn:incoming>license_stored_yes</bpmn:incoming>
            <bpmn:outgoing>store event license created</bpmn:outgoing>
        </bpmn:parallelGateway>
        <bpmn:choreographyTask initiatingParticipantRef="ras_client" name="Store Event"
id="store event license task">
            <bpmn:incoming>store_event_license_created</bpmn:incoming>
            <bpmn:participantRef>ras_client</bpmn:participantRef>
```

```
            <bpmn:participantRef>store event sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_event_req_message_flow2</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store_event_rsp_message_flow2</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:endEvent name="RAS_End" id="end_successful" >
            <bpmn:incoming>ras_end_successful</bpmn:incoming>
        </bpmn:endEvent>
        <bpmn:sequenceFlow sourceRef="RAS_Start" targetRef="authenticate_user_task"
id="authenticate_user_ES"/>
        <bpmn:sequenceFlow sourceRef="authenticate_user_task" targetRef="gw_user_authenticate"
name="" id="is_user_authenticated"/>
        <bpmn:sequenceFlow sourceRef="gw_user_authenticate" targetRef="create_content_task"
name="Yes" id="user_authenticate_yes"/>
        <bpmn:sequenceFlow sourceRef="gw_user_authenticate" targetRef="end_not_authenticated"
name="No" id="user_authenticate_no"/>
        <bpmn:sequenceFlow sourceRef="create_content_task" targetRef="gw_content_created" name=""
id="is content created"/>
        <bpmn:sequenceFlow sourceRef="gw_content_created" targetRef="identify_content_task"
name="Yes" id="content_created_yes"/>
        <bpmn:sequenceFlow sourceRef="gw content created" targetRef="end not created" name="No"
id="content_created_no"/>
        <bpmn:sequenceFlow sourceRef="identify_content_task" targetRef="gw_content_identified"
name="" id="is content identified"/>
        <bpmn:sequenceFlow sourceRef="gw_content_identified" targetRef="store_content_task"
name="Yes" id="content_identified_yes"/>
        <bpmn:sequenceFlow sourceRef="gw content identified" targetRef="end not identified" name="No"
id="content_identified_no"/>
        <bpmn:sequenceFlow sourceRef="gw_content_store" targetRef="par_gw_store_event_content"
name="Yes" id="content stored yes"/>
        <bpmn:sequenceFlow sourceRef="gw_content_store" targetRef="end_not_stored" name="No"
id="content_stored_no"/>
        <bpmn:sequenceFlow sourceRef="store content task" targetRef="gw content store" name=""
id="is_content_stored"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_content" targetRef="create_license_task"
name="" id="event and content stored"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_content"
targetRef="store_event_content_task" name="" id="store_event_content"/>
        <bpmn:sequenceFlow sourceRef="gw license created" targetRef="store license task" name="Yes"
id="license_created_yes"/>
        <bpmn:sequenceFlow sourceRef="gw_license_created" targetRef="end_no_license" name="No"
id="license_created_no"/>
        <bpmn:sequenceFlow sourceRef="create_license_task" targetRef="gw_license_created" name=""
id="is_license_created"/>
        <bpmn:sequenceFlow sourceRef="gw_license_stored" targetRef="par_gw_store_event_license"
name="Yes" id="license_stored_yes"/>
        <bpmn:sequenceFlow sourceRef="gw_license_stored" targetRef="end_no_stored_license" name="No"
id="license_stored_no"/>
        <bpmn:sequenceFlow sourceRef="store license task" targetRef="gw license stored" name=""
id="event_and_license_stored"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_license"
targetRef="store event license task" name="" id="store event license created"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_license" targetRef="end_successful"
id="ras_end_successful"/>
    </bpmn:choreography>
</bpmn:definitions>
```

6.4.3 Application of the methodology to the buyer use case

Step 1. Narrative description

In this use case, a user wants to purchase some multimedia content on sale. To do so, she connects to a web-based application which is the front-end to a MPEG-M compliant system which provides her with the functionality requested, that is, content search, licensing capabilities and event reporting, among others.

Step 2. Identification of ESs and ASs present in the AS

The ESs to be used are: Authenticate User, Search Content, Create License, Store License, Store Event and Authorize User.

Step 3. Workflow textual description

The service workflow associated to the Buy and Consume Content aggregated service (BAC) is as follows:

Steps	Service invoked	Client + Message	Service Provider + Message	Description
1.	Authenticate User (ES)	BAC Client `AuthenticateUserRequest`	Authenticate User SP	*BAC Client* tries to authenticate in the system.
2.	Authenticate User (ES)	BAC Client	Authenticate User SP `AuthenticateUserResponse`	*Authenticate User SP* tries to authenticate BAC Client with the information provided. If response is negative, AS stops here.
3.	Search Content (ES)	BAC Client `SearchContentRequest`	Search Content SP	*BAC Client* makes a search over offered content.
4.	Search Content (ES)	BAC Client	Search Content SP `SearchContentResponse`	*Search Content SP* makes a search with the requirements given by the BAC Client. If this operation is not successful, AS stops here.
5.	Create License (ES)	BAC Client `CreateLicenseRequest`	Create License SP	*BAC Client* provides the required information to create some licenses over content found in step 4.
6.	Create License (ES)	BAC Client	Create License SP `CreateLicenseResponse`	*Create License SP* creates the license(s) with the information sent from client. If this operation is not successful, AS stops here.
7.	Store License (ES)	BAC Client `StoreLicenseRequest`	Store License SP	*BAC Client* requests license storage.
8.	Store License (ES)	BAC Client	Store License SP `StoreLicenseResponse`	*Store License SP* stores the licenses sent by BAC Client. If this operation is not successful, AS stops here.
9.	Store Event (ES)	BAC Client `StoreEventRequest`	Store Event SP	*BAC Client* requests event to be stored. (This step is done in parallel with step 11).

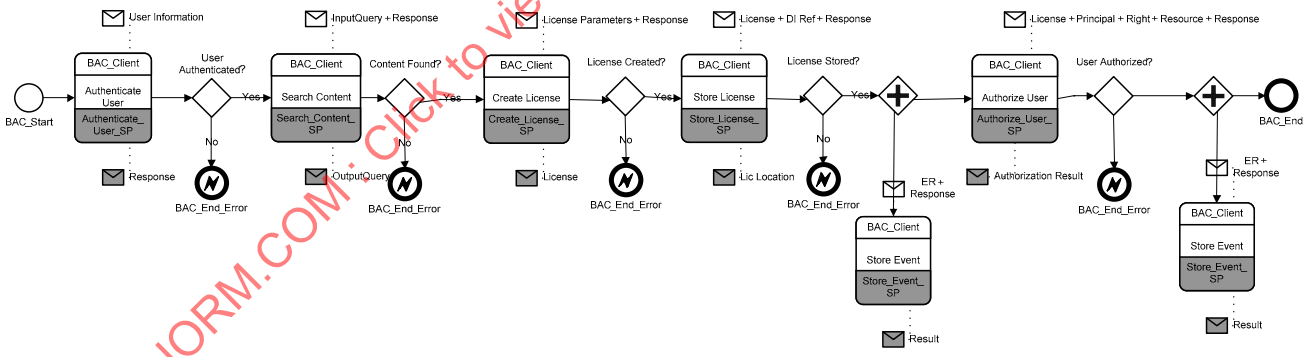
Steps	Service invoked	Client + Message	Service Provider + Message	Description
10.	Store Event (ES)	BAC Client	Store Event SP `StoreEventResponse`	*Store Event SP* stores an event informing of the licenses stored.
11.	Authorize User (ES)	BAC Client `AuthorizeUserRequest`	Authorize User SP	*BAC Client* requests authorization for content consumption.
12.	Authorize User (ES)	BAC Client	Authorize User SP `AuthorizeUserResponse`	*Authorize User SP* if user is authorized. If this operation is not successful, AS stops here.
13.	Store Event (ES)	BAC Client `StoreEventRequest`	Store Event SP	*BAC Client* requests event to be stored.
14.	Store Event (ES)	BAC Client	Store Event SP `StoreEventResponse`	*Store Event SP* stores an event informing of the content consumption.

Step 4. Formal description of the service workflow

The methodology does not define which representation has to be used for the formal description of the service workflow, just that a diagram and an optional XML serialization should be provided. In this example, BPMN 2.0 is used for the formal description of the proposed AS (see Clause 7 for details).

The description is as follows:

BPMN 2.0 Diagram



BPMN 2.0 XML serialization

```
<bpmn:definitions id="BAC_Choreography" name="BAC_Choreography" targetNamespace="bac"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:mpegm="urn:mpeg:mpegM:service:01-service-NS:2012">
    <bpmn:choreography id="BAC_Service" name="Buy And Consume content">
        <bpmn:participant name="BAC Client" id="bac client"/>
        <bpmn:participant name="Authenticate_User_SP" id="authenticate_user_sp"/>
        <bpmn:participant name="Search_Content_SP" id="search_content_sp"/>
        <bpmn:participant name="Create_License_SP" id="create_license_sp"/>
```

```
        <bpmn:participant name="Store License SP" id="store license sp"/>
        <bpmn:participant name="Store_Event_SP" id="store_event_sp"/>
        <bpmn:participant name="Authorize_User_SP" id="authorize_user_sp"/>
        <bpmn:messageFlow sourceRef="bac client" targetRef="authenticate user sp"
id="authenticate_user_req_message_flow" messageRef="mpegm:authenticate_user_request"/>
        <bpmn:messageFlow sourceRef="authenticate_user_sp" targetRef="bac_client"
id="authenticate user rsp message flow" messageRef="mpegm:authenticate user response"/>
        <bpmn:messageFlow sourceRef="bac_client" targetRef="search_content_sp"
id="search_content_req_message_flow" messageRef="mpegm:search_content_request"/>
        <bpmn:messageFlow sourceRef="search_content_sp" targetRef="bac_client"
id="search_content_rsp_message_flow" messageRef="mpegm:search_content_response"/>
        <bpmn:messageFlow sourceRef="bac_client" targetRef="create_license_sp"
id="create_license_req_message_flow" messageRef="mpegm:create_license_request"/>
        <bpmn:messageFlow sourceRef="create_license_sp" targetRef="bac_client"
id="create_license_rsp_message_flow" messageRef="mpegm:create_license_response"/>
        <bpmn:messageFlow sourceRef="bac_client" targetRef="store_license_sp"
id="store license req message flow" messageRef="mpegm:store license request"/>
        <bpmn:messageFlow sourceRef="store_license_sp" targetRef="bac_client"
id="store_license_rsp_message_flow" messageRef="mpegm:store_license_response"/>
        <bpmn:messageFlow sourceRef="bac client" targetRef="store event sp"
id="store_event_req_message_flow1" messageRef="mpegm:store_event_request"/>
        <bpmn:messageFlow sourceRef="store_event_sp" targetRef="bac_client"
id="store event rsp message flow1" messageRef="mpegm:store event response"/>
        <bpmn:messageFlow sourceRef="bac_client" targetRef="authorize_user_sp"
id="authorize_user_req_message_flow1" messageRef="mpegm:authorize_user_request"/>
        <bpmn:messageFlow sourceRef="authorize user sp" targetRef="bac client"
id="authorize_user_rsp_message_flow1" messageRef="mpegm:authorize_user_response"/>
        <bpmn:messageFlow sourceRef="bac_client" targetRef="store_event_sp"
id="store event req message flow2" messageRef="mpegm:store event request"/>
        <bpmn:messageFlow sourceRef="store_event_sp" targetRef="bac_client"
id="store_event_rsp_message_flow2" messageRef="mpegm:store_event_response"/>
        <bpmn:startEvent id="BAC Start" name="BAC Start">
            <bpmn:outgoing>authenticate_user_ES</bpmn:outgoing>
        </bpmn:startEvent>
        <bpmn:choreographyTask initiatingParticipantRef="bac client" name="Authenticate User"
id="authenticate_user_task">
            <bpmn:incoming>authenticate_user_ES</bpmn:incoming>
            <bpmn:outgoing>is user authenticated</bpmn:outgoing>
            <bpmn:participantRef>bac_client</bpmn:participantRef>
            <bpmn:participantRef>authenticate_user_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>authenticate_user_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>authenticate_user_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="User Authenticated?"
id="gw_user_authenticate">
            <bpmn:incoming>is_user_authenticated</bpmn:incoming>
            <bpmn:outgoing>user_authenticate_yes</bpmn:outgoing>
            <bpmn:outgoing>user authenticate no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="BAC_End_Error" id="end_not_authenticated">
            <bpmn:incoming>user authenticate no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="bac client" name="Search Content"
id="search_content_task">
            <bpmn:incoming>user_authenticate_yes</bpmn:incoming>
            <bpmn:outgoing>is content found</bpmn:outgoing>
            <bpmn:participantRef>bac_client</bpmn:participantRef>
```

```
            <bpmn:participantRef>search content sp</bpmn:participantRef>
            <bpmn:messageFlowRef>search_content_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>search_content_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="Content Found?"
id="gw_content_found">
            <bpmn:incoming>is content found</bpmn:incoming>
            <bpmn:outgoing>content_found_yes</bpmn:outgoing>
            <bpmn:outgoing>content_found_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="BAC_End_Error" id="end_not_found">
            <bpmn:incoming>content_found_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="bac_client" name="Create License"
id="create_license_task">
            <bpmn:incoming>content found yes</bpmn:incoming>
            <bpmn:outgoing>is_license_created</bpmn:outgoing>
            <bpmn:participantRef>bac_client</bpmn:participantRef>
            <bpmn:participantRef>create license sp</bpmn:participantRef>
            <bpmn:messageFlowRef>create_license_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>create_license_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="License Created?"
id="gw_license_created">
            <bpmn:incoming>is license created</bpmn:incoming>
            <bpmn:outgoing>license_created_yes</bpmn:outgoing>
            <bpmn:outgoing>license_created_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="BAC_End_Error" id="end_no_license">
            <bpmn:incoming>license_created_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:choreographyTask initiatingParticipantRef="bac_client" name="Store License"
id="store license task">
            <bpmn:incoming>license_created_yes</bpmn:incoming>
            <bpmn:outgoing>event_and_license_stored</bpmn:outgoing>
            <bpmn:participantRef>bac client</bpmn:participantRef>
            <bpmn:participantRef>store_license_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_license_req_message_flow</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store_license_rsp_message_flow</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="License Stored?"
id="gw_license_stored">
            <bpmn:incoming>event_and_license_stored</bpmn:incoming>
            <bpmn:outgoing>license_stored_yes</bpmn:outgoing>
            <bpmn:outgoing>license_stored_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="BAC_End_Error" id="end_no_stored_license">
            <bpmn:incoming>license_stored_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:parallelGateway gatewayDirection="Unspecified" name="Store Event and Store License"
id="par gw store event license">
            <bpmn:incoming>license_stored_yes</bpmn:incoming>
            <bpmn:outgoing>store_event_license_created</bpmn:outgoing>
            <bpmn:outgoing>authorize user license created</bpmn:outgoing>
        </bpmn:parallelGateway>
```

```
        <bpmn:choreographyTask initiatingParticipantRef="bac client" name="Store Event"
id="store_event_license_task">
            <bpmn:incoming>store_event_license_created</bpmn:incoming>
            <bpmn:participantRef>bac client</bpmn:participantRef>
            <bpmn:participantRef>store_event_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_event_req_message_flow1</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store event rsp message flow1</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:choreographyTask initiatingParticipantRef="bac_client" name="Authorize User"
id="authorize_user_task">
            <bpmn:incoming>authorize_user_license_created</bpmn:incoming>
            <bpmn:outgoing>is_user_authorised</bpmn:outgoing>
            <bpmn:participantRef>bac_client</bpmn:participantRef>
            <bpmn:participantRef>authorize_user_sp</bpmn:participantRef>
            <bpmn:messageFlowRef>authorize_user_req_message_flow1</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>authorize_user_req_message_flow1</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:exclusiveGateway gatewayDirection="Unspecified" name="User Authorized?"
id="gw_authorize_user">
            <bpmn:incoming>is user authorised</bpmn:incoming>
            <bpmn:outgoing>user_authorized_yes</bpmn:outgoing>
            <bpmn:outgoing>user_authorized_no</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:endEvent name="BAC_End_Error" id="end_no_authorized">
            <bpmn:incoming>user_authorized_no</bpmn:incoming>
            <bpmn:errorEventDefinition/>
        </bpmn:endEvent>
        <bpmn:parallelGateway gatewayDirection="Unspecified" name="Auth User Store Event"
id="par gw authorize store event">
            <bpmn:incoming>user_authorized_yes</bpmn:incoming>
            <bpmn:outgoing>BAC_completed</bpmn:outgoing>
            <bpmn:outgoing>authorize user store event</bpmn:outgoing>
        </bpmn:parallelGateway>
        <bpmn:choreographyTask initiatingParticipantRef="bac_client" name="Store Event"
id="store event auth user task">
            <bpmn:incoming>authorize_user_store_event</bpmn:incoming>
            <bpmn:participantRef>bac_client</bpmn:participantRef>
            <bpmn:participantRef>store event sp</bpmn:participantRef>
            <bpmn:messageFlowRef>store_event_req_message_flow2</bpmn:messageFlowRef>
            <bpmn:messageFlowRef>store_event_rsp_message_flow2</bpmn:messageFlowRef>
        </bpmn:choreographyTask>
        <bpmn:endEvent name="BAC_End" id="end_successful">
            <bpmn:incoming>BAC_completed</bpmn:incoming>
        </bpmn:endEvent>
        <bpmn:sequenceFlow sourceRef="authenticate_user_task" targetRef="gw_user_authenticate"
name="" id="is_user_authenticated"/>
        <bpmn:sequenceFlow sourceRef="gw_user_authenticate" targetRef="search_content_task"
name="Yes" id="user authenticate yes"/>
        <bpmn:sequenceFlow sourceRef="gw_user_authenticate" targetRef="end_not_authenticated"
name="No" id="user_authenticate_no"/>
        <bpmn:sequenceFlow sourceRef="gw content found" targetRef="create license task" name="Yes"
id="content_found_yes"/>
        <bpmn:sequenceFlow sourceRef="gw_content_found" targetRef="end_not_found" name="No"
id="content found no"/>
        <bpmn:sequenceFlow sourceRef="search_content_task" targetRef="gw_content_found" name=""
id="is_content_found"/>
        <bpmn:sequenceFlow sourceRef="gw license created" targetRef="store license task" name="Yes"
id="license_created_yes"/>
```

```
        <bpmn:sequenceFlow sourceRef="gw license created" targetRef="end no license" name="No"
id="license_created_no"/>
        <bpmn:sequenceFlow sourceRef="create_license_task" targetRef="gw_license_created" name=""
id="is license created"/>
        <bpmn:sequenceFlow sourceRef="gw_license_stored" targetRef="par_gw_store_event_license"
name="Yes" id="license_stored_yes"/>
        <bpmn:sequenceFlow sourceRef="gw license stored" targetRef="end no stored license" name="No"
id="license_stored_no"/>
        <bpmn:sequenceFlow sourceRef="store_license_task" targetRef="gw_license_stored" name=""
id="event_and_license_stored"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_license"
targetRef="store_event_license_task" name="" id="store_event_license_created"/>
        <bpmn:sequenceFlow sourceRef="BAC_Start" targetRef="authenticate_user_task" name=""
id="authenticate_user_ES"/>
        <bpmn:sequenceFlow sourceRef="par_gw_store_event_license" targetRef="authorize_user_task"
name="" id="authorize_user_license_created"/>
        <bpmn:sequenceFlow sourceRef="authorize user task" targetRef="gw authorize user" name=""
id="is_user_authorised"/>
        <bpmn:sequenceFlow sourceRef="gw_authorize_user" targetRef="par_gw_authorize_store_event"
name="Yes" id="user authorized yes"/>
        <bpmn:sequenceFlow sourceRef="gw_authorize_user" targetRef="end_no_authorized" name="No"
id="user_authorized_no"/>
        <bpmn:sequenceFlow sourceRef="par gw authorize store event" targetRef="end successful"
name="" id="BAC_completed"/>
        <bpmn:sequenceFlow sourceRef="par_gw_authorize_store_event"
targetRef="store event auth user task" name="" id="authorize user store event"/>
    </bpmn:choreography>
</bpmn:definitions>
```


7 Formal representation of service workflows

7.1 Introduction

As stated in 6.2, a formal representation of service workflow of aggregated services is required. This representation has to describe the relationship between the different services (elementary, external or aggregated) comprising the AS as well as participants and messages interchanged.

Several workflow and process representation languages exist that provide both graphical and XML representations of service workflows. These languages provide support for representation of the required information to completely describe the service workflow of an aggregated service. The aim of this part of ISO/IEC 23006 is not to provide a new representation for service workflow but to indicate which is the minimum required to describe it and the relationship with ESs defined in ISO/IEC 23006-4:2012. With this purpose, 7.2, 7.3 and 7.4 describe how ASs can be represented using BPMN 2.0, although this is not the only way to formally describe ASs. 7.2 and 7.3 describe two different mechanisms for the definition of both BPMN diagrams and XML serializations based on `bpmn:collaboration` and `bpmn:choreography`, respectively.

7.2 BPMN 2.0 Representation of aggregated services using bmpn:collaboration

BPMN is a possible formal representation of service aggregation. This subclause specifies how to use the OMG BPMN 2.0 XML representation for Service aggregation using `bpmn:collaboration` element. OMG BPMN 2.0 specifies a notation to represent processes, which is a formalism to describe Service workflows. Using this notation, it is possible to represent temporal events, associations, and precedences that are combining Services into an Aggregated Service. From this point of view, Service aggregation is seen as an instance of a process described using BPMN 2.0.

BPMN 2.0 specifies a graphical notation and an XML representation for processes. This part of ISO/IEC 23006 uses both representations for the definition of Service workflows. ISO/IEC 23006-4:2012 uses the BPMN 2.0 representation to define the workflows of all Elementary Services. The workflow of an Aggregated Service may be defined through a BPMN 2.0 representation.

A service aggregation can express a process flow realizing a specific task as well as a new service. The service provider can expose Elementary Services or Aggregated Services, constructed from several other Services.

Since service aggregation is a key point of the MSPT, BPMN use is adequate as it allows a deep description of service interactions. Moreover, many different aggregation topologies (not only a serial version of aggregation) and contacts among services can be quite easily illustrated employing this method.

When using `bpmn:collaboration`, the BPMN 2.0 XML representation of an Aggregated Service must contain all of the following elements:

— A `bpmn:collaboration` that identifies the participants (i.e., client, SP of the Aggregated Service, and optionally SPs of the included Services) and the message flows between them.

— A `bpmn:process` representing a BPMN Public Process for the workflow of the SP of the Aggregated Service.

If the Aggregated Service defines its own messages, the BPMN 2.0 XML representation must additionally contain the following elements:

— `bpmn:message` elements for all messages exchanged between client and SP. Each `bpmn:message` references a `bpmn:itemDefinition`.

— `bpmn:itemDefinition` elements that reference the actual XML messages of the protocol data format.

The following rules apply to the `bpmn:collaboration` element:

The `bpmn:collaboration` shall specify the `id` attribute in order to allow references from the `bpmn:process` of the SP. The `name` attribute shall be the name of the Aggregated Service (e.g., "TV-Multimedia Processing").

The `bpmn:participant` elements shall specify the name attributes (e.g., "Client" for the client, "TV-Multimedia Processing SP" for the SP of the Aggregated Service, and "Process Content SP - TVM Get Configuration" for the SP of the included Service). The `bpmn:participant` representing the SP of the Aggregated Service must reference the `bpmn:process` of the Aggregated Service through the `processRef` attribute. Each `bpmn:participant` representing an SP of an included Service must reference the `bpmn:process` of the included Service through the `processRef` attribute. The `bpmn:participant` representing the client shall specify the `id` attribute in order to enable further Service aggregation.

The `bpmn:messageFlow` elements must reference through their `sourceRef` and `targetRef` attributes the `bpmn:process` for the client, the BPMN Flow Nodes (i.e., Events, Tasks, Gateways) within the `bpmn:process` for the SP of the Aggregated Service, and/or the `bpmn:process` of the SP of an included Service. If a `bpmn:messageFlow` element models the message flow of a composing Service, then the `bpmn:messageFlow` element must specify the `id` attribute, which must match the `id` attribute of the `bpmn:messageFlow` element defined in the BPMN representation of the composing Service. Unless specified otherwise, the `bpmn:messageFlow` elements must reference the corresponding `bpmn:messages` through the `messageRef` attribute. For example, if the `bpmn:messageFlow` element of an Elementary Service has the `id` attribute value "create_content_request_message_flow", then an Aggregated Service modelling this message flow must also set the `id` attribute of the corresponding `bpmn:messageFlow` element to "create_content_request_message_flow".

The following rules apply to the bpmn:process element:

The bpmn:process must specify the id attribute in order to allow references from the bpmn:participant of the bpmn:collaboration and to enable further Service aggregation. The processType attribute shall be set to "Public". The bpmn:collaboration of the Elementary Service shall be referenced through the definitionalCollaborationRef attribute.

The bpmn:process shall contain a bpmn:startEvent, which is triggered by the first message flow from the client to the SP of the Aggregated Service. Further BPMN Flow Nodes (i.e., Events, Tasks, Gateways, Sub-Processes), and the bpmn:sequenceFlow elements connecting them, specify a high-level workflow for the SP of the Aggregated Service.

The following rules apply to the bpmn:message elements:

The bpmn:message must specify the id attribute in order to allow references from the bpmn:messageFlow. The name attribute shall be the name of the protocol message of the Aggregated Service. The corresponding bpmn:itemDefinition must be referenced through the itemRef attribute.

The following rules apply to the bpmn:itemDefinition elements:

The bpmn:itemDefinition must specify the id attribute in order to allow references from the bpmn:message elements. The XML element of the actual protocol message of the Aggregated Service must be referenced through the structureRef attribute.

7.3 BPMN 2.0 Representation of aggregated services using bpmn:choreography

BPMN is a possible formal representation of service aggregation. This subclause specifies how to use the OMG BPMN 2.0 XML representation for Service aggregation based on bpmn:choreography element. OMG BPMN 2.0 specifies a notation to represent choreographies, which is a formalism to describe Service workflows (in fact, BPMN provides several mechanisms for presenting service workflows, some of them equivalent). Using this notation, it is possible to represent temporal events, associations, and precedences that are combining Services into an Aggregated Service. From this point of view, Service aggregation is seen as an instance of a choreography described using BPMN 2.0.

BPMN 2.0 specifies both, a graphical notation and an XML representation for processes. This part of ISO/IEC 23006 uses both representations for the definition of Service workflows. ISO/IEC 23006-4:2012 uses the BPMN 2.0 representation to define the workflows of all Elementary Services. The workflow of an Aggregated Service may be defined through a BPMN 2.0 representation.

A service aggregation can express a process flow realizing a specific task as well as a new service. The service provider can expose Elementary Services or Aggregated Services, constructed from several other Services.

Since service aggregation is a key point of the MSPT, BPMN use is adequate as it allows a deep description of service interactions. Moreover, many different aggregation topologies (not only a serial version of aggregation) and contacts among services can be quite easily illustrated employing this method.

When using bpmn:choreography, the BPMN 2.0 XML representation of an Aggregated Service must contain all of the following elements:

— A bpmn:choreography that identifies the participants (i.e., client, SP of the Aggregated Service, and SPs of the included Services) and the message flows between them.

— A bpmn:choreographyTask representing a BPMN task inside the choreography of the complete Aggregated Service. It makes reference to the participants and message flows already defined at choreography level.

If the Aggregated Service defines its own messages, the BPMN 2.0 XML representation must additionally contain the following elements:

— `bpmn:message` elements for all messages exchanged between client and SP. Each `bpmn:message` references a `bpmn:itemDefinition`.

— `bpmn:itemDefinition` elements that reference the actual XML messages of the protocol data format.

The following rules apply to the `bpmn:choreography` element:

The `bpmn:choreography` shall specify the `id` attribute in order to allow references from other elements inside the BPMN serialization. The `name` attribute shall be the name of the Aggregated Service (e.g., "Register And Sell Content").

The `bpmn:participant` elements shall specify the name attributes (e.g., "RAS_Client" for the client and "Authenticate User SP" for the SP of the included Service). Each `bpmn:participant` shall specify the `id` attribute in order to facilitate reference.

The `bpmn:messageFlow` elements must reference through their `sourceRef` and `targetRef` attributes the `bpmn:participant` for each BPMN Flow Nodes (i.e, Events, Tasks, Gateways) within the `bpmn:choreography` of the Aggregated Service. If a `bpmn:messageFlow` element models the message flow of a composing Service, then the `bpmn:messageFlow` element must specify the `id` attribute, which must match the `id` attribute of the `bpmn:messageFlow` element defined in the BPMN representation of the composing Service. Unless specified otherwise, the `bpmn:messageFlow` elements must reference the corresponding `bpmn:messages` through the `messageRef` attribute. For example, if the `bpmn:messageFlow` element of an Elementary Service has the `id` attribute value "create_content_request_message_flow", then an Aggregated Service modelling this message flow must also set the `id` attribute of the corresponding `bpmn:messageFlow` element to "create_content_request_message_flow".

The following rules apply to the `bpmn:choreography` element:

The `bpmn:choreography` must specify the `id` attribute in order to allow references to enable further Service aggregation.

The `bpmn:choreography` shall contain a `bpmn:startEvent`, which is triggered by the first message flow from the client to the first Service (Elementary or Aggregated) inside the Aggregated Service. Further BPMN Flow Nodes (i.e. Events, Tasks, Gateways), and the `bpmn:sequenceFlow` elements connecting them, specify a high-level workflow for the SP of the Aggregated Service.

7.4 Representation of Service Types

This subclause defines a BPMN 2.0 extension for defining the Service Type of a generic Elementary Service (e.g., Process Content) within the context of service aggregation.

The binding of the workflow of a generic Elementary Service to a Service Type is accomplished through the extensibility mechanism of BPMN 2.0.

The XML Schema definition of the BPMN 2.0 extension for Service Types is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:bpmnext2="urn:mpeg:mpegM:schema:06-bpmn-ext-st-NS:2012"
xmlns:mpegmb="urn:mpeg:mpegM:schema:01-base-NS:2012"
targetNamespace="urn:mpeg:mpegM:schema:06-bpmn-ext-st-NS:2012"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
  <import namespace="urn:mpeg:mpegM:schema:01-service-NS:2011"
schemaLocation="mpeg-m-base.xsd"/>
  <element name="ServiceTypeEntity" type="mpegmb:ServiceTypeEntityType"/>
</schema>
```

Semantics of the `bpmnext2:ServiceTypeEntity`:

Name	*Definition*
`bpmnext2:ServiceTypeEntity`	Specifies the Service Type of a generic Elementary Service. The `bpmnext2:ServiceTypeEntity` element shall only be used as an extension to `bpmn:participant` and `bpmn:messageFlow` elements. For the `bpmn:participant` element, the `bpmnext2:ServiceTypeEntity` element specifies the Service Type of the generic Elementary Service for which the `bpmn:participant` acts as the SP. For the `bpmn:messageFlow` element, the `bpmnext2:ServiceTypeEntity` element specifies the Service Type of the generic Elementary Service to which the `bpmn:messageFlow` belongs.

If the Elementary Service has a Service Type, then the following additional rules apply:

The surrounding `bpmn:definitions` root element must specify a `bpmn:extension` element. The `mustUnderstand` attribute of the `bpmn:extension` element must be set to "true". The `definition` attribute must reference the `bpmnext2:ServiceTypeEntity` element.

Within the BPMN 2.0 representation of the Aggregated Service, each `bpmn:participant` element that represents a SP of a generic Elementary Service with a Service Type must contain a `bpmn:extensionElements` element which contains exactly one `bpmnext2:ServiceTypeEntity` element. The `bpmnext2:ServiceTypeEntity` element must convey the same Service Type as the Service Instance Declaration of that Elementary Service.

Within the BPMN 2.0 representation of the Aggregated Service, each `bpmn:messageFlow` element that belongs to a generic Elementary Service with a Service Type must contain a `bpmn:extensionElements` element which contains exactly one `bpmnext2:ServiceTypeEntity` element. The `bpmnext2:ServiceTypeEntity` element must convey the same Service Type as the Service Instance Declaration of that Elementary Service.

Annex A
(informative)

MPEG-M Elementary and Aggregated Services Registration Procedure

A.1 General information

The MPEG-M registration mechanism supports both registration and request of Elementary and Aggregated Services definitions and its associated information. The registration is performed by the MPEG-M Elementary and Aggregated Services Registration Authority ("MPEG-M Services RA", in short). ISO/IEC JTC 1 maintains a list of registration authorities at http://www.iso.org/iso/maintenance_agencies. Developers of new MPEG-M services may submit their Elementary or Aggregated Services compliant information to the MPEG-M Services RA for registration. Other users may utilize the MPEG-M Services RA services to find registered Elementary or Aggregated Services information. Submitters are responsible for keeping their ES or AS information available online if they want to allow usage from external independent applications.

The MPEG-M Services RA shall verify the syntactic correctness of the provided information and shall perform regular checks to verify that the registered information and links keep valid. Furthermore, the RA shall provide for easy identification of Elementary or Aggregated Services.

A.2 Access to the MPEG-M Elementary and Aggregated Services Registration Authority

The MPEG-M Services RA web site shall provide forms for Registration of Applications, Search of Registered Elementary and Aggregated Services Information and Request for Replacement. All the cases use a form that includes a structure encoded in XML for organizing the information associated to the ES.

The form of Registration of Applications shall include the following information:

- provider information that contains name, description, contact information, and provider capability,
- For the case of Elementary Service registration, the information to be provided contains description, name, protocol, service, formal representation, which includes the service workflow diagram and its corresponding XML serialization.
- For the case of Aggregated Service registration, the information to be provided contains the results of the application of the methodology for the definition of Aggregated Services, as described in 6.2 of this part of ISO/IEC 23006.

The form of Search of Registered Elementary and Aggregated Services shall include the following information:

- Elementary or Aggregated Service name.

The form of Request of Replacement shall include the following information:

- provider information,
- For the case of Elementary Service replacement, the information to be provided contains description, name, protocol, service, formal representation, which includes the service workflow diagram and its corresponding XML serialization to be replaced.

- For the case of Aggregated Service replacement, the information to be provided contains the results of the application of the methodology for the definition of Aggregated Services, as described in 6.2 of this part of ISO/IEC 23006.

A.3 Review and response to applicants

This subclause defines the process for the MPEG-M Elementary and Aggregated Services Registration Authority to review and respond to applications to ensure fairness.

In the case of Registration of Applications, a technical review committee is set up to review the applications. This committee is composed of Registration Authority members and may use software tools to help in the application evaluation process. The applications are answered by the review committee in less than three weeks after the application submission date.

The review committee accepts or rejects the Elementary or Aggregated Service, based on the acceptance criteria in A.4. If accepted, the new registered ES or AS is allocated an identifier (ID). The ID shall then be used for management of ES or AS.

Once the application has been reviewed and accepted, the RA notifies the user of a positive or negative response to the registration request. The response to the user shall include a short explanation of the results of the technical review. A negative response may be appealed if the registrant believes that there was an error made in the rejection, or when further information is required to clarify issues or concerns. If the registrant requests additional review beyond the Authority's process, the case may be submitted to ISO/IEC JTC1/SC29/WG11 for review, which might require the MPEG-M Services RA to re-evaluate the application.

The form of result for Registration of Applications shall include the following information:

- identification information for Elementary or Aggregated Service,
- status, warning, or exception information,
- information about the details of the result.

The form of result for Search of Registered Elementary or Aggregated Services shall include the following information:

- Elementary or Aggregated Service identification information,
- information about the location where the Elementary or Aggregated Service information can be obtained,
- information about the provider of the requested Elementary or Aggregated Service.

The form of result for Request of Replacement shall include the following information:

- Elementary or Aggregated Service identification information,
- status, warning, or exception information.

A.4 Acceptance criteria for registration application

An application is always accepted unless one or more of these situations occur:

- The applicant is not eligible, i.e. the submitter is not an identifiable person or organization.
- An approved, registered Elementary or Aggregated Service already exists that contain identical content than the submission.

- The information in the application is incomplete or incomprehensible.
- The technical description is not sufficient.
- The justification for inclusion in the register is not adequate. The candidate Elementary or Aggregated Service should demonstrate it provides an appropriate definition of Elementary or Aggregated Service and give examples of use cases when relevant.
- The Authority considers that there is not enough originality in the proposed Elementary Service which could easily be implemented with an existing ES. The same applies to AS.
- The submission contains errors or it is not compliant with the normative MPEG-M standard.

A.5 Assignment of identifiers

The review process and syntax ensures that the assigned ID is unique within the RA and that the same ID is not assigned to another ES or AS. After the assignment has been made, the ID and associated information shall be included in the RA registry and the MPEG-M Services RA shall inform the user of the assignment.

The MPEG-M Services RA definition shall be recorded in the registry at the time when the ID is assigned.

A.6 Maintenance

For the purpose of maintenance of the registry, the MPEG-M Services RA shall implement mechanisms for maintaining the integrity of registry including adequate backup for retaining records.

An ID owner may update the information associated to Elementary or Aggregated Services through a Request of Replacement.

A user can request information about registered Elementary or Aggregated Services through a Search of Registered Elementary or Aggregated Services.

Annex B
(informative)

Create and Identify Content Aggregated Service

This annex gives an example of an Aggregated Service, which is named *Create and Identify Content* (*CIC*). The Aggregated Service is composed of two Elementary Services: the *Create Content* ES and the *Identify Content* ES. The application of the methodology described in 6.2 is presented below.

Step 1. Narrative description

In this use case, a content creator wants to create and identify some multimedia content she has created (music score, video, photo or a combination of them).

Step 2. Identification of ESs and ASs present in the AS

The ESs to be used are: Create Content and Identify Content. Both of them are defined in ISO/IEC 23006-*Step 3. Workflow textual description*

The service workflow associated to the Create and Identify Content (CIC) is as follows:

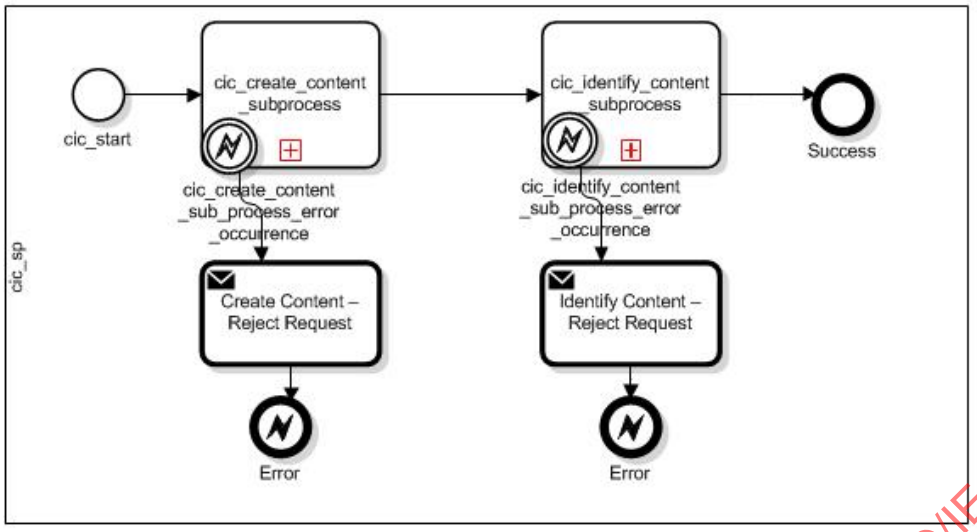
Steps	Service invoked	Client + Message	Service Provider + Message	Description
1.	Create Content (ES)	CIC Client `CreateContentRequest`	Create Content SP	*CIC Client* provides the required information to create some content. This information may include content metadata, resources and licenses (offers at this step).
2.	Create Content (ES)	CIC Client	Create Content SP `CreateContentResponse`	*Create Content SP* creates the content with the information sent from client. If this operation is not successful, AS stops here.
3.	Identify Content (ES)	CIC Client `IdentifyContentRequest`	Identify Content SP	*CIC Client* requests an identifier for the newly created content.
4.	Identify Content (ES)	CIC Client	Identify Content SP `IdentifyContentResponse`	*Identify Content SP* generates an identifier for the content. If this operation is not successful, AS stops here.

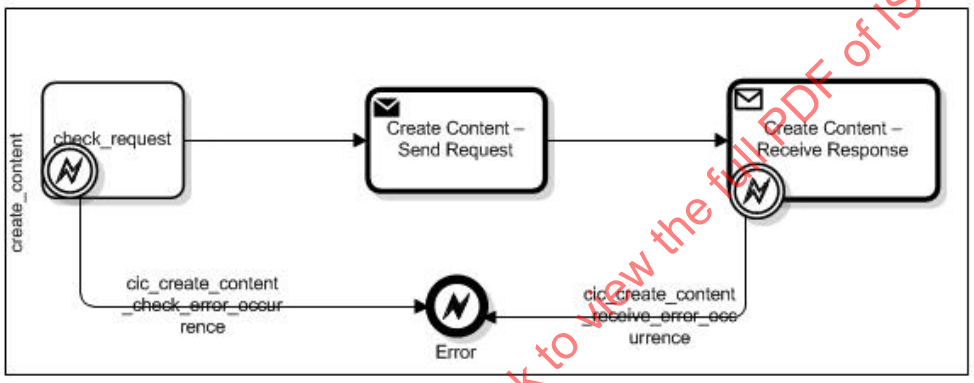
Step 4. Formal description of the service workflow

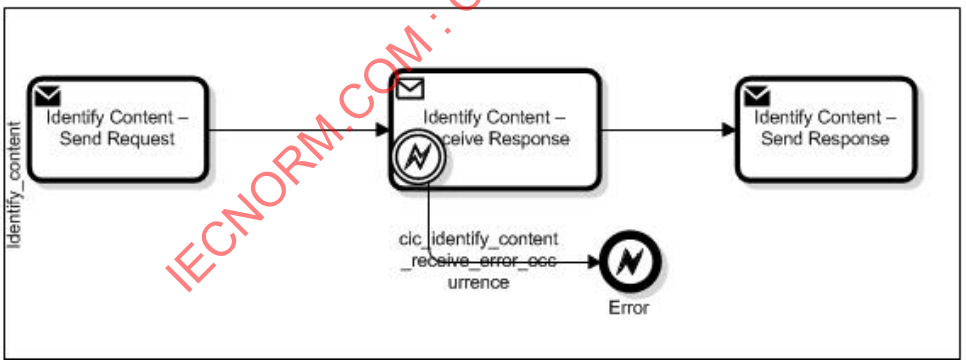
The methodology does not define which representation has to be used for the formal description of the service workflow, just that a diagram and an optional XML serialization should be provided. In this example, BPMN 2.0 is used for the formal description of the proposed AS (see Clause 7 for details).

This description is as follows:

BPMN 2.0 Diagram







BPMN 2.0 XML serialization

The BPMN XML serialization of the aggregated service is shown below. The interaction between the Services and the exception handling are represented.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions targetNamespace="urn:example:service:cic"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:mpegm="urn:mpeg:mpegM:service:01-service-NS:2012"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL ../omg/BPMN20.xsd">
  <bpmn:import importType="http://www.omg.org/spec/BPMN/20100524/MODEL"
location="ES.bpmn.xml" namespace="urn:mpeg:bpmn:01-service-NS:2012"/>

  <!-- ********************************************************** -->
  <!--  Collaboration and Process for Create and Identify Content -->
  <!-- ********************************************************** -->

  <bpmn:collaboration id="cic_collaboration" name="Create and Identify Content">
    <bpmn:participant id="cic_client" name="Client"/>
    <bpmn:participant name="Create and Identify Content SP" processRef="cic_sp"/>

    <bpmn:participant name="Create Content SP" processRef="mpegm:create_content_sp"/>
    <bpmn:participant name="Identify Content SP" processRef="mpegm:identify_content_sp"/>

    <!-- Initial Request -->
    <bpmn:messageFlow id="cic_request_message_flow" name="CIC Request MessageFlow"
sourceRef="cic_client" targetRef="cic_start" messageRef="mpegm:create_content_request"/>

    <!-- Create Content SubProcess MessageFlows -->
    <bpmn:messageFlow id="create_content_request_message_flow" name="CIC Create Content
Request MessageFlow" sourceRef="cic_create_content_send_request"
targetRef="mpegm:create_content_sp" messageRef="mpegm:create_content_request"/>
    <bpmn:messageFlow id="create_content_reject_message_flow" name="CIC Create Content
Reject MessageFlow" sourceRef="mpegm:create_content_sp"
targetRef="cic_create_content_receive_response"
messageRef="mpegm:create_content_response"/>
    <bpmn:messageFlow id="create_content_response_message_flow" name="CIC Create Content
Response MessageFlow" sourceRef="mpegm:create_content_sp"
targetRef="cic_create_content_receive_response"
messageRef="mpegm:create_content_response"/>

    <!-- Failure1 -->
    <bpmn:messageFlow id="cic_reject_message_flow1" name="CIC Reject MessageFlow 1"
sourceRef="cic_create_content_reject_request" targetRef="cic_client"
messageRef="mpegm:create_content_response"/>

    <!-- Identify Content SubProcess MessageFlows -->
    <bpmn:messageFlow id="identify_content_request_message_flow" name="CIC Identify
Content Request MessageFlow" sourceRef="cic_identify_content_send_request"
targetRef="mpegm:identify_content_sp" messageRef="mpegm:identify_content_request"/>
    <bpmn:messageFlow id="identify_content_reject_message_flow" name="CIC Identify Content
Reject MessageFlow" sourceRef="mpegm:identify_content_sp"
targetRef="cic_identify_content_receive_response"
messageRef="mpegm:identify_content_response"/>
    <bpmn:messageFlow id="identify_content_response_message_flow" name="CIC Identify
Content Response MessageFlow" sourceRef="mpegm:identify_content_sp"
targetRef="cic_identify_content_receive_response"
messageRef="mpegm:identify_content_response"/>
```

```
    <!-- Failure 2 -->
    <bpmn:messageFlow id="cic_reject_message_flow2" name="CIC Reject MessageFlow 2"
sourceRef="cic_identify_content_reject_request" targetRef="cic_client"
messageRef="mpegm:create_content_response"/>

    <!-- Final Response -->
    <bpmn:messageFlow id="cic_response_message_flow" name="CIC Response MessageFlow"
sourceRef="cic_identify_content_send_response" targetRef="cic_client"
messageRef="mpegm:create_content_response"/>
  </bpmn:collaboration>

  <bpmn:process id="cic_sp" processType="Public"
definitionalCollaborationRef="cic_collaboration">
    <bpmn:startEvent id="cic_start">
      <bpmn:messageEventDefinition/>
    </bpmn:startEvent>

    <!-- Create Content SubProcess -->
    <bpmn:sequenceFlow sourceRef="cic_start" targetRef="cic_create_content_sub_process"/>
    <bpmn:subProcess id="cic_create_content_sub_process">
      <!-- Check Request -->
      <bpmn:task id="cic_create_content_check_request" name="Check Request"/>
      <bpmn:boundaryEvent id="cic_create_content_check_error_occurrence"
attachedToRef="cic_create_content_check_request">
        <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
      </bpmn:boundaryEvent>

      <!-- Check Request Error -->
      <bpmn:sequenceFlow sourceRef="cic_create_content_check_error_occurrence"
targetRef="cic_create_content_sub_process_error_end"/>

      <!-- Check Request OK -->
      <bpmn:sequenceFlow sourceRef="cic_create_content_check_request"
targetRef="cic_create_content_send_request"/>
      <bpmn:sendTask id="cic_create_content_send_request" name="Create Content - Send
Request"/>
      <bpmn:sequenceFlow sourceRef="cic_create_content_send_request"
targetRef="cic_create_content_receive_response"/>
      <bpmn:receiveTask id="cic_create_content_receive_response" name="Create Content -
Receive Response"/>
      <bpmn:boundaryEvent id="cic_create_content_receive_error_occurrence"
attachedToRef="cic_create_content_receive_response">
        <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
      </bpmn:boundaryEvent>

      <!-- Receive Response Error -->
      <bpmn:sequenceFlow sourceRef="cic_create_content_receive_error_occurrence"
targetRef="cic_create_content_sub_process_error_end"/>
      <bpmn:endEvent id="cic_create_content_sub_process_error_end" name="Error">
        <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
        <!-- End -->
      </bpmn:endEvent>

      <!-- No response sent to client yet. -->
    </bpmn:subProcess>
    <bpmn:boundaryEvent id="cic_create_content_sub_process_error_occurrence"
attachedToRef="cic_create_content_sub_process">
      <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
    </bpmn:boundaryEvent>

    <!-- Create Content Error -->
    <bpmn:sequenceFlow sourceRef="cic_create_content_sub_process_error_occurrence"
targetRef="cic_create_content_reject_request"/>
```

```
    <bpmn:sendTask id="cic_create_content_reject_request" name="Create Content - Reject
Request"/>
    <bpmn:sequenceFlow sourceRef="cic_create_content_reject_request"
targetRef="cic_create_content_error_end"/>
    <bpmn:endEvent id="cic_create_content_error_end" name="Error">
      <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
      <!-- End -->
    </bpmn:endEvent>

    <!-- Identify Content SubProcess -->
    <bpmn:sequenceFlow sourceRef="cic_create_content_sub_process"
targetRef="cic_identify_content_sub_process"/>
    <bpmn:subProcess id="cic_identify_content_sub_process">
      <bpmn:sendTask id="cic_identify_content_send_request" name="Identify Content - Send
Request"/>
      <bpmn:sequenceFlow sourceRef="cic_identify_content_send_request"
targetRef="cic_identify_content_receive_response"/>
      <bpmn:receiveTask id="cic_identify_content_receive_response" name="Identify Content
- Receive Response"/>
      <bpmn:boundaryEvent id="cic_identify_content_receive_error_occurrence"
attachedToRef="cic_identify_content_receive_response">
        <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
      </bpmn:boundaryEvent>

      <!-- Receive Response Error -->
      <bpmn:sequenceFlow sourceRef="cic_identify_content_receive_error_occurrence"
targetRef="cic_identify_content_sub_process_error_end"/>
      <bpmn:endEvent id="cic_identify_content_sub_process_error_end" name="Error">
        <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
        <!-- End -->
      </bpmn:endEvent>

      <!-- Receive Response OK -->
      <bpmn:sequenceFlow sourceRef="cic_identify_content_receive_response"
targetRef="cic_identify_content_send_response"/>
      <bpmn:sendTask id="cic_identify_content_send_response" name="Identify Content - Send
Response"/>
    </bpmn:subProcess>
    <bpmn:boundaryEvent id="cic_identify_content_sub_process_error_occurrence"
attachedToRef="cic_identify_content_sub_process">
      <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
    </bpmn:boundaryEvent>

    <!-- Identify Content Error -->
    <bpmn:sequenceFlow sourceRef="cic_identify_content_sub_process_error_occurrence"
targetRef="cic_identify_content_reject_request"/>
    <bpmn:sendTask id="cic_identify_content_reject_request" name="Identify Content -
Reject Request"/>
    <bpmn:sequenceFlow sourceRef="cic_identify_content_reject_request"
targetRef="cic_identify_content_error_end"/>
    <bpmn:endEvent id="cic_identify_content_error_end" name="Error">
      <bpmn:eventDefinition xsi:type="bpmn:tErrorEventDefinition"/>
      <!-- End -->
    </bpmn:endEvent>

    <bpmn:sequenceFlow sourceRef="cic_identify_content_sub_process"
targetRef="cic_success"/>
    <bpmn:endEvent id="cic_success" name="Success"/>
  </bpmn:process>
</bpmn:definitions>
```

Annex C
(informative)

TV-Multimedia Processing Aggregated Service

C.1 Aggregated Service Definition

This Annex gives an example of an Aggregated Service, which is named *TV-Multimedia (TVM) Processing*. The Aggregated Service is composed of three Elementary Services: the *Process Content* ES with the additionally defined Service Type *TVM Get Configuration*, again the *Process Content* ES with the additionally defined Service Type *TVM Configure*, and again the *Process Content* ES with the additionally defined Service Type *TVM Command*. The application of the methodology described in subclause 6.2 is presented below.

Step 1. Narrative description

In this use case, a user wants to use a Device for processing content according to the capabilities of the Process Content ES defined in subclause 5.16.1 of ISO/IEC 23006-4. However, the processing is performed in a multi-step workflow: in the first step, the current configuration of the Device is retrieved; in the second step, the user can update the configuration; in the third step, the actual commands (e.g., "START", "STOP", etc.) are applied in order to perform the actual processing. Steps two and three can be repeated multiple times.

Step 2. Identification of ESs and ASs present in the AS

The only ES used is Process Content as defined in ISO/IEC 23006-4:2012.

TV-Multimedia Processing defines three interconnected Service Types for Process Content:

— TVM Get Configuration

— TVM Configure

— TVM Command

NOTE The three Service Types correspond to the three steps of controlling a TVM Processor. In the first step (i.e., the TVM Get Configure Service Type), the current configuration is retrieved from the TVM Processor. If desired, the configuration can be updated in the second step (i.e., the TVM Configure Service Type). In the third step (i.e., TVM Command Service Type), the actual commands (e.g., "START", "STOP", etc.) are applied in order to perform the actual TVM function. The steps two and three can be repeated multiple times.

Step 3. Workflow textual description

The service workflow associated to the TV-Multimedia Processing is as follows:

Steps	Service invoked	Client + Message	Service Provider + Message	Description
1.	Process Content (ES) with Service Type *TVM Get Configuration*	Client `ProcessContentRequest`	Process Content SP	*Client* requests the current configuration of the Device.

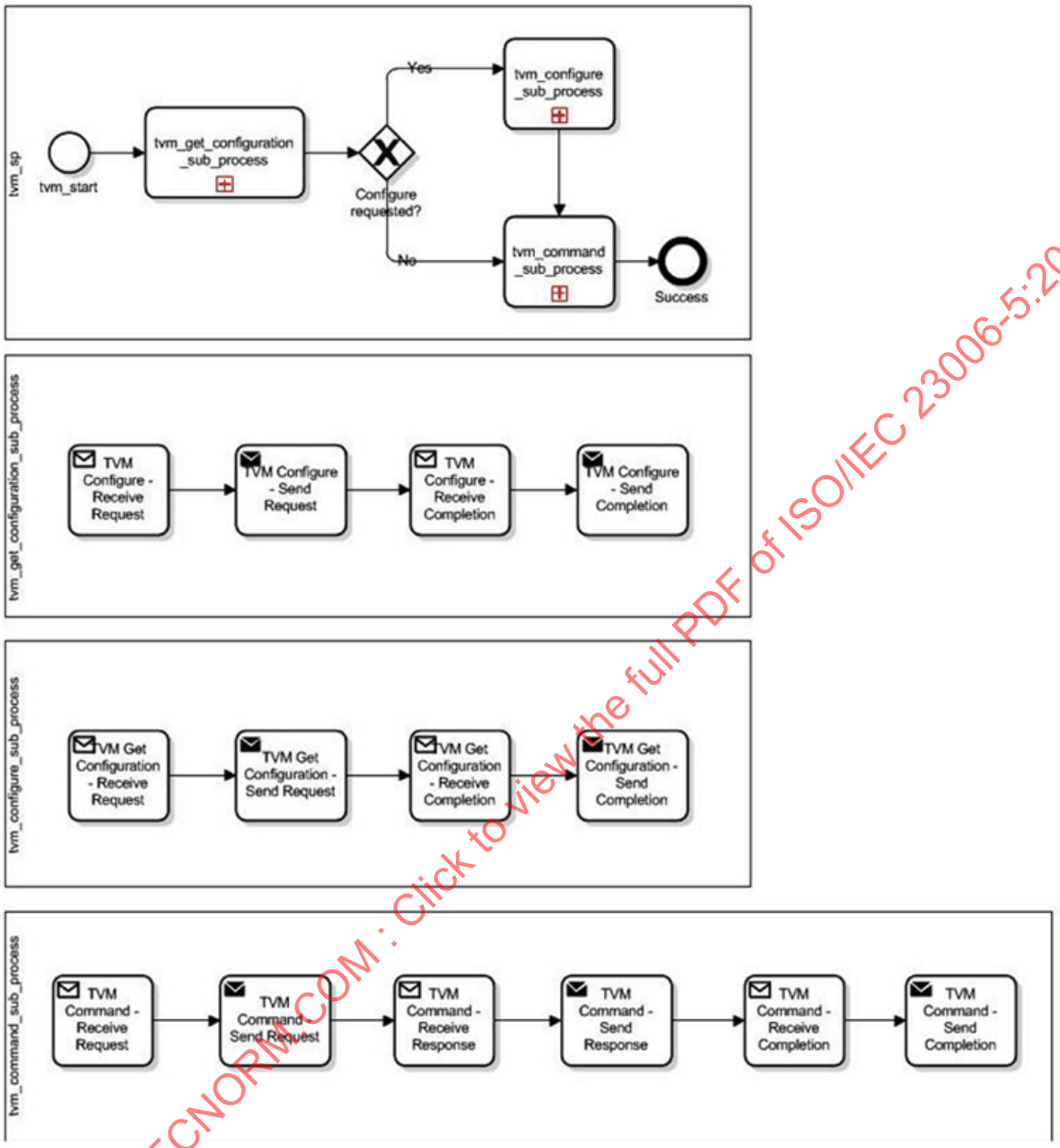
Steps	Service invoked	Client + Message	Service Provider + Message	Description
2.	Process Content (ES) with Service Type *TVM Get Configuration*	Client	Process Content SP `ProcessContentCompletion`	*Process Content SP* sends the current configuration. If this operation is not successful, AS stops here.
3.				If the *Client* does not want to update the configuration, it continues with step 6,
4.	Process Content (ES) with Service Type *TVM Configure*	Client `ProcessContentRequest`	Process Content SP	*Client* sends updated configuration.
5.	Process Content (ES) with Service Type *TVM Configure*	Client	Process Content SP `ProcessContentCompletion`	*Process Content SP* applies the updated configuration. If this operation is not successful, AS stops here.
6.	Process Content (ES) with Service Type *TVM Command*	Client `ProcessContentRequest`	Process Content SP	*Client* signals the command that shall be executed (e.g., "START" for starting the processing with the applied configuration).
7.	Process Content (ES) with Service Type *TVM Command*	Client	Process Content SP `ProcessContentResponse`	*Process Content SP* executes the command and signals whether the command can be executed. If this operation is not successful, AS stops here.
8.				*Client* may go back to step 4 in order to update the configuration.
9.	Process Content (ES) with Service Type *TVM Command*	Client	Process Content SP `ProcessContentCompletion`	*Process Content SP* indicates that this request has been completed after either the source stream has ended or the *Client* has sent another `ProcessContentRequest` message of Service Type *TVM Command* with new commands.

Step 4. Formal description of the service workflow

The methodology does not define which representation has to be used for the formal description of the service workflow, just that a diagram and an optional XML serialization should be provided. In this example, BPMN 2.0 is used for the formal description of the proposed AS (see Clause 7 for details).

This description is as follows:

BPMN 2.0 Diagram



BPMN 2.0 XML serialization

The BPMN representation of the aggregated service is shown below.

NOTE 1 The `bpmn:subProcess` elements for the TVM Get Configuration and TVM Configure Service Types define only tasks for request and response messages. Only the `bpmn:subProcess` for the TVM Command Service Type (with the id attribute value "`tvm_command_sub_process`") defines tasks for request, response, and completion messages.

NOTE 2 Message flows in case of failures are not handled explicitly in this BPMN representation.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<bpmn:definitions targetNamespace="urn:mpeg:mpegM:bpmn:02-as-tvm-NS:2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:mpegm="urn:mpeg:mpegM:schema:01-service-NS:2011"
xmlns:sid="urn:mpeg:mpegM:schema:04-sid-NS:2011" xmlns:bpmnext2="urn:mpeg:mpegM:schema:06-
bpmn-ext-st-NS:2011" xmlns:tvm="urn:mpeg:mpegM:service-type:03-tvm-NS:2011"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL ../omg/BPMN20.xsd
urn:mpeg:mpegM:schema:06-bpmn-ext-st-NS:2011 ../bpmn-ext-st.xsd urn:mpeg:mpegM:service-
type:03-tvm-NS:2011 ../service_types/mpeg-m-ST-tvm.xsd">
  <bpmn:import importType="http://www.omg.org/spec/BPMN/20100524/MODEL"
location="../bpmn/ES.bpmn.xml" namespace="urn:mpeg:mpegM:bpmn:01-service-NS:2012"/>

  <bpmn:extension definition="bpmnext2:ServiceTypeEntity" mustUnderstand="true">
    <bpmn:documentation>Used in bpmn:participant to specify the Service Type of a Process
Content bpmn:process and in bpmn:messageFlow to specify the Service Type of a Process
Content message.</bpmn:documentation>
  </bpmn:extension>
  <!-- ************************************************************** -->
  <!-- Collaboration and Process for TV-Multimedia Processing         -->
  <!-- ************************************************************** -->
  <bpmn:collaboration id="tvm_collaboration" name="TV-Multimedia Processing">
    <bpmn:participant id="tvm_client" name="Client"/>
    <bpmn:participant name="TV-Multimedia Processing SP" processRef="tvm_sp"/>
    <bpmn:participant name="Process Content SP - TVM Get Configuration"
processRef="mpegm:process_content_sp">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <sid:ServiceType id="tvm_get_configuration_service_type">
            <sid:Identifier>urn:mpeg:mpegM:service-type:03-tvm-
NS:2011:TVMGetConfiguration</sid:Identifier>
            <sid:Name>TVM Get Configuration</sid:Name>

<sid:RequestParametersType>tvm:TVMGetConfigurationRequestParametersType</sid:RequestParame
tersType>

<sid:CompletionParametersType>tvm:TVMGetConfigurationCompletionParametersType</sid:Complet
ionParametersType>
          </sid:ServiceType>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:participant>
    <bpmn:participant name="Process Content SP - TVM Configure"
processRef="mpegm:process_content_sp">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <sid:ServiceType id="tvm_configure_service_type">
            <sid:Identifier>urn:mpeg:mpegM:service-type:03-tvm-
NS:2011:TVMConfigue</sid:Identifier>
            <sid:Name>TVM Configure</sid:Name>

<sid:RequestParametersType>tvm:TVMConfigureRequestParametersType</sid:RequestParametersTyp
e>
          </sid:ServiceType>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:participant>
    <bpmn:participant name="Process Content SP - TVM Command"
processRef="mpegm:process_content_sp">
```

```
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <sid:ServiceType id="tvm_command_service_type">
            <sid:Identifier>urn:mpeg:mpegM:service-type:03-tvm-
NS:2011:TVMCommand</sid:Identifier>
            <sid:Name>TVM Command</sid:Name>

<sid:RequestParametersType>tvm:TVMCommandRequestParametersType</sid:RequestParametersType>
          </sid:ServiceType>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:participant>

    <bpmn:messageFlow id="tvm_get_configuration_request_message_flow"
sourceRef="tvm_client" targetRef="tvm_start" messageRef="mpegm:process_content_request">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_get_configuration_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_request_message_flow"
sourceRef="tvm_get_configuration_send_request" targetRef="mpegm:process_content_sp"
messageRef="mpegm:process_content_request">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_get_configuration_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_completion_message_flow"
sourceRef="mpegm:process_content_sp" targetRef="tvm_get_configuration_receive_completion"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_get_configuration_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_get_configuration_completion_message_flow"
sourceRef="tvm_get_configuration_send_completion" targetRef="tvm_client"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_get_configuration_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_configure_request_message_flow" sourceRef="tvm_client"
targetRef="tvm_configure_receive_request" messageRef="mpegm:process_content_request">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_configure_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_request_message_flow"
sourceRef="tvm_configure_send_request" targetRef="mpegm:process_content_sp"
messageRef="mpegm:process_content_request">
```

```
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_configure_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_completion_message_flow"
sourceRef="mpegm:process_content_sp" targetRef="tvm_configure_receive_completion"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_configure_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_configure_completion_message_flow"
sourceRef="tvm_configure_send_completion" targetRef="tvm_client"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_configure_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_command_request_message_flow" sourceRef="tvm_client"
targetRef="tvm_command_receive_request" messageRef="mpegm:process_content_request">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_request_message_flow"
sourceRef="tvm_command_send_request" targetRef="mpegm:process_content_sp"
messageRef="mpegm:process_content_request">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="process_content_response_message_flow"
sourceRef="mpegm:process_content_sp" targetRef="tvm_command_receive_response"
messageRef="mpegm:process_content_response">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_command_response_message_flow"
sourceRef="tvm_command_send_response" targetRef="tvm_client"
messageRef="mpegm:process_content_response">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
```

```
    <bpmn:messageFlow id="process_content_completion_message_flow"
sourceRef="mpegm:process_content_sp" targetRef="tvm_command_receive_completion"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
    <bpmn:messageFlow id="tvm_command_completion_message_flow"
sourceRef="tvm_command_send_completion" targetRef="tvm_client"
messageRef="mpegm:process_content_completion">
      <bpmn:extensionElements>
        <bpmnext2:ServiceTypeEntity>
          <mpegm:ServiceTypeRef>#tvm_command_service_type</mpegm:ServiceTypeRef>
        </bpmnext2:ServiceTypeEntity>
      </bpmn:extensionElements>
    </bpmn:messageFlow>
  </bpmn:collaboration>

  <bpmn:process isExecutable="false" id="tvm_sp"
definitionalCollaborationRef="tvm_collaboration">
    <bpmn:startEvent id="tvm_start" name="Start">
      <bpmn:messageEventDefinition/>
    </bpmn:startEvent>
    <bpmn:sequenceFlow sourceRef="tvm_start"
targetRef="tvm_get_configuration_sub_process"/>
    <bpmn:subProcess id="tvm_get_configuration_sub_process">
      <bpmn:receiveTask id="tvm_get_configuration_receive_request" name="TVM Get
Configuration - Receive Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_get_configuration_receive_request"
targetRef="tvm_get_configuration_send_request"/>
      <bpmn:sendTask id="tvm_get_configuration_send_request" name="TVM Get Configuration -
Send Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_get_configuration_send_request"
targetRef="tvm_get_configuration_receive_completion"/>
      <bpmn:receiveTask id="tvm_get_configuration_receive_completion" name="TVM Get
Configuration - Receive Completion"/>
      <bpmn:sequenceFlow sourceRef="tvm_get_configuration_receive_completion"
targetRef="tvm_get_configuration_send_completion"/>
      <bpmn:sendTask id="tvm_get_configuration_send_completion" name="TVM Get
Configuration - Send Completion"/>
    </bpmn:subProcess>
    <bpmn:sequenceFlow sourceRef="tvm_get_configuration_sub_process"
targetRef="tvm_is_configure_requested"/>
    <bpmn:eventBasedGateway id="tvm_is_configure_requested" name="Configure requested?"/>
    <bpmn:sequenceFlow sourceRef="tvm_is_configure_requested"
targetRef="tvm_configure_sub_process" name="Yes"/>
    <bpmn:sequenceFlow sourceRef="tvm_is_configure_requested"
targetRef="tvm_command_sub_process" name="No"/>
    <bpmn:subProcess id="tvm_configure_sub_process">
      <bpmn:receiveTask id="tvm_configure_receive_request" name="TVM Configure - Receive
Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_configure_receive_request"
targetRef="tvm_configure_send_request"/>
      <bpmn:sendTask id="tvm_configure_send_request" name="TVM Configure - Send Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_configure_send_request"
targetRef="tvm_configure_receive_completion"/>
```

```
      <bpmn:receiveTask id="tvm_configure_receive_completion" name="TVM Configure -
Receive Completion"/>
      <bpmn:sequenceFlow sourceRef="tvm_configure_receive_completion"
targetRef="tvm_configure_send_completion"/>
      <bpmn:sendTask id="tvm_configure_send_completion" name="TVM Configure - Send
Completion"/>
    </bpmn:subProcess>
    <bpmn:sequenceFlow sourceRef="tvm_configure_sub_process"
targetRef="tvm_command_sub_process"/>
    <bpmn:subProcess id="tvm_command_sub_process">
      <bpmn:loopCharacteristics xsi:type="bpmn:tStandardLoopCharacteristics"/>
      <bpmn:receiveTask id="tvm_command_receive_request" name="TVM Command - Receive
Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_command_receive_request"
targetRef="tvm_command_send_request"/>
      <bpmn:sendTask id="tvm_command_send_request" name="TVM Command - Send Request"/>
      <bpmn:sequenceFlow sourceRef="tvm_command_send_request"
targetRef="tvm_command_receive_response"/>
      <bpmn:receiveTask id="tvm_command_receive_response" name="TVM Command - Receive
Response"/>
      <bpmn:sequenceFlow sourceRef="tvm_command_receive_response"
targetRef="tvm_command_send_response"/>
      <bpmn:sendTask id="tvm_command_send_response" name="TVM Command - Send Response"/>
      <bpmn:sequenceFlow sourceRef="tvm_command_send_response"
targetRef="tvm_command_receive_completion"/>
      <bpmn:receiveTask id="tvm_command_receive_completion" name="TVM Command - Receive
Completion"/>
      <bpmn:sequenceFlow sourceRef="tvm_command_receive_completion"
targetRef="tvm_command_send_completion"/>
      <bpmn:sendTask id="tvm_command_send_completion" name="TVM Command - Send
Completion"/>
    </bpmn:subProcess>
    <bpmn:sequenceFlow sourceRef="tvm_command_sub_process" targetRef="tvm_success"/>
    <bpmn:endEvent id="tvm_success" name="Success"/>
  </bpmn:process>
</bpmn:definitions>
```


C.2 Syntax of Service Type data format

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpegmb="urn:mpeg:mpegM:schema:01-base-
NS:2011" xmlns:tvm="urn:mpeg:mpegM:service-type:03-tvm-NS:2012"
targetNamespace="urn:mpeg:mpegM:service-type:03-tvm-NS:2012"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <import namespace="urn:mpeg:mpeg7:schema:2004"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
7_schema_files/mpeg7-v3.xsd"/>
  <import namespace="urn:mpeg:mpegM:schema:01-base-NS:2011" schemaLocation="../mpeg-m-
base.xsd"/>
  <import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
schemaLocationhttp://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
21_schema_files/dia-2nd/UED-2nd.xsd"/>
  <!-- ################################################ -->
  <!--  TVMGetConfigurationRequestType                  -->
  <!-- ################################################ -->
```

```
  <complexType name="TVMGetConfigurationRequestType">
    <complexContent>
      <extension base="mpegmb:RequestParametersBaseType">
        <sequence>
          <element name="TVMFunctionIdentifier" type="tvm:TVMFunctionIdentifierType"
minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  TVMGetConfigurationCompletionType               -->
  <!-- ################################################ -->
  <complexType name="TVMGetConfigurationCompletionType">
    <complexContent>
      <extension base="mpegmb:CompletionParametersBaseType">
        <sequence>
          <element name="ServiceTypeEntity" type="mpegmb:ServiceTypeEntityType"/>
          <element name="Configuration" type="mpegmb:RequestParametersBaseType"/>
          <element name="DeviceDescription" type="dia:TerminalsType" minOccurs="0"/>
        </sequence>
        <attribute name="tvmFunctionName" type="string"/>
        <attribute name="tvmFunctionVersion" type="string"/>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  TVMConfigureRequestType                         -->
  <!-- ################################################ -->
  <complexType name="TVMConfigureRequestType">
    <complexContent>
      <extension base="mpegmb:RequestParametersBaseType">
        <sequence>
          <element name="TVMFunctionIdentifier" type="tvm:TVMFunctionIdentifierType"/>
          <element name="Configuration" type="mpegmb:RequestParametersBaseType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  TVMCommandRequestType                           -->
  <!-- ################################################ -->
  <complexType name="TVMCommandRequestType">
    <complexContent>
      <extension base="mpegmb:RequestParametersBaseType">
        <sequence>
          <element name="TVMFunctionIdentifier" type="tvm:TVMFunctionIdentifierType"/>
          <element name="CommandCode" type="tvm:CommandCodeType"/>
          <element name="CommandArgument" type="string" minOccurs="0"/>
          <element name="ScheduledTime" type="dateTime" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  General Definitions: Capabilities               -->
  <!-- ################################################ -->
  <!-- Definition of TVMFunctionIdentifierType -->
  <simpleType name="TVMFunctionIdentifierType">
```

```
    <restriction base="anyURI"/>
  </simpleType>
  <!-- Definition of NetworkInterfaceCapabilitiesType -->
  <complexType name="NetworkInterfaceCapabilitiesType">
    <complexContent>
      <extension base="dia:TerminalCapabilityBaseType">
        <sequence>
          <element name="NetworkInterface" type="tvm:NetworkInterfaceType" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- Definition of ContentIOCapabilitiesBaseType -->
  <complexType name="ContentIOCapabilitiesBaseType" abstract="true">
    <complexContent>
      <extension base="dia:TerminalCapabilityBaseType">
        <sequence>
          <element name="ContentIOType" type="mpeg7:ControlledTermUseType" minOccurs="0"
maxOccurs="unbounded">
            <annotation>
              <documentation xml:lang="en">Suggested ClassificationScheme:
"urn:mpeg:mpegM:cs:02-ST-tvm-NS:2011:TVMContentIOTypeCS"</documentation>
            </annotation>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- Definition of SourceCapabilitiesType -->
  <complexType name="SourceCapabilitiesType">
    <complexContent>
      <extension base="tvm:ContentIOCapabilitiesBaseType"/>
    </complexContent>
  </complexType>
  <!-- Definition of TargetCapabilitiesType -->
  <complexType name="TargetCapabilitiesType">
    <complexContent>
      <extension base="tvm:ContentIOCapabilitiesBaseType"/>
    </complexContent>
  </complexType>
  <!-- ################################################ -->
  <!--  General Definitions: Command                    -->
  <!-- ################################################ -->
  <!-- Definition of CommandCodeType -->
  <simpleType name="CommandCodeType">
    <restriction base="NMTOKEN">
      <enumeration value="INIT"/>
      <enumeration value="START"/>
      <enumeration value="STOP"/>
      <enumeration value="APPLY"/>
      <enumeration value="RESET"/>
    </restriction>
  </simpleType>
  <!-- ################################################ -->
  <!--  General Definitions: Network                    -->
  <!-- ################################################ -->
  <!-- Definition of NetworkInterfaceType -->
  <complexType name="NetworkInterfaceType">
```

```
    <simpleContent>
      <extension base="tvm:CIDRType">
        <attribute name="name" type="NCName"/>
      </extension>
    </simpleContent>
  </complexType>
  <!-- Definition of CIDRType -->
  <simpleType name="CIDRType">
    <restriction base="string"/>
  </simpleType>
</schema>
```

C.3 Semantics of Service Type data format

Semantics of the `TVMGetConfigurationRequestType`:

Name	*Definition*
`TVMGetConfigurationRequestType`	Parameters for the `mpegm:ProcessContentRequest` message of Service Type "TVM Get Configuration". `TVMGetConfigurationRequestType` extends `mpegmb:RequestParametersBaseType`.
`TVMFunctionIdentifier`	Identifies the TVM Processor and the actual TVM Function.

Semantics of the `TVMGetConfigurationCompletionType`:

Name	*Definition*
`TVMGetConfigurationCompletionType`	Parameters for the `mpegm:ProcessContentCompletion` message of Service Type "TVM Get Configuration". `TVMGetConfigurationCompletionType` extends `mpegmb:CompletionParametersBaseType`.
`ServiceTypeEntity`	Service Type of the Process Content elementary service instance, typically indicated via identifier.
`Configuration`	Current configuration of the requested TVM Function. The type must comply with the `mpegmb:RequestParametersType` of the `ServiceTypeEntity`.
`DeviceDescription`	Capabilities of the TVM Processor. The capabilities are represented as MPEG-21 `dia:TerminalCapabilities`. Recommended types of Terminal Capabilities are the `tvm:NetworkInterfaceCapabilitiesType`, `tvm:SourceCapabilitiesType`, `dia:CodecCapabilitiesType`, and `tvm:TargetCapabilitiesType`.

Name	*Definition*
tvmFunctionName	Descriptive name of the TVM Function.
tvmFunctionVersion	Indicates the version of the TVM Function.

Semantics of the TVMConfigureRequestType:

Name	*Definition*
TVMConfigureRequestType	Parameters for the mpegm:ProcessContentRequest message of Service Type "TVM Configure". TVMConfigureRequestType extends mpegmb:RequestParametersBaseType.
TVMFunctionIdentifier	Identifies the TVM Processor and the actual TVM Function.
Configuration	Updated configuration for the specified TVM Function. The type must comply with the mpegmb:RequestParametersType of the ServiceTypeEntity defined in the TVMGetConfigurationCompletionParametersType of the preceding mpegm:ProcessContentCompletion message of Service Type TVM Get Configuration.

Semantics of the TVMCommandRequestType:

Name	*Definition*
TVMCommandRequestType	Parameters for the mpegm:ProcessContentRequest message of Service Type "TVM Command". TVMCommandRequestType extends mpegmb:RequestParametersBaseType.
TVMFunctionIdentifier	Identifies the TVM Processor and the actual TVM Function.
CommandCode	Command that shall be applied to the TVM Processor.
CommandArgument	Application-speicific arguments for the command.
ScheduledTime	Time when the command shall be applied. If omitted, the command must be applied immediately.

Semantics of the `TVMFunctionIdentifierType`:

Name	*Definition*
`TVMFunctionIdentifierType`	Type for identification of the TVM Processor and the actual TVM Function.

Semantics of the `NetworkInterfaceCapabilitiesType`:

Name	*Definition*
`NetworkInterfaceCapabilitiesType`	Top-level type for representation of network interfaces. `NetworkInterfaceCapabilitiesType` extends `dia:TerminalCapabilityBaseType`.
`NetworkInterface`	Identifies a network interface and the end-point to which the device is currently connected.

Semantics of the `ContentIOCapabilitiesType`:

Name	*Definition*
`ContentIOCapabilitiesType`	Abstract top-level type for the representation of supported content input/output types. `ContentIOCapabilitiesType` extends `dia:TerminalCapabilityBaseType`.
`ContentIOType`	Indicates a supported content input/output type. An example of a Classification Scheme is the TVMContentIOTypeCS (`urn:mpeg:mpegM:cs:02-ST-tvm-NS:2012:TVMContentIOTypeCS`) defined in C.5.1

Semantics of the `SourceCapabilitiesType`:

Name	*Definition*
`SourceCapabilitiesType`	Top-level type for the representation of supported input types. `SourceCapabilitiesType` extends `dia:TerminalCapabilityBaseType`.

Semantics of the `TargetCapabilitiesType`:

Name	*Definition*
`TargetCapabilitiesType`	Top-level type for the representation of supported output types. `TargetCapabilitiesType` extends `dia:TerminalCapabilityBaseType`.

Semantics of the `CommandCodeType`:

Name	*Definition*
`CommandCodeType`	Command code that shall be executed by the SP. Possible values are shown in Table 1.

Semantics of the `NetworkInterfaceType`:

Name	*Definition*
`NetworkInterfaceType`	Network interface to which a device is connected. `NetworkInterfaceType` extends `CIDRType`.
`ID`	Identifier of the network interface, unique per device (e.g., "eth0").

Semantics of the `CIDRType`:

Name	*Definition*
`CIDRType`	Classless Inter-Domain Routing (CIDR) notation for IP addresses (e.g., "192.0.2.0/24" or "2001:db8::/32").

Table 1 — Possible values of the `CommandCodeType`.

Command Code	Definition
INIT	Initialize the device (if applicable).
START	Start processing.
STOP	Stop processing.
APPLY	Apply new configuration to a running operation.
RESET	Reset the device (e.g., for error recovery).