Special Paper
방송공학회논문지 제27권 제7호, 2022년 12월 (JBE Vol. 27, No. 7, December 2022)
https://doi.org/10.5909/JBE.2022.27.7.1034
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

Mission Description Structure for Autonomous Collaboration of Media Things

Jonghoon Chun[a)] and Sang-Kyun Kim[a)‡]

Abstract

Internet of Media Things (IoMT) aims to provide intelligent media services to users by connecting (media) things to each other and exchanging media data. The mission given by the user is automatically executed, and intelligent media service can be provided to the user. To achieve this, ISO/IEC 23093-5 started standardization work on the mission data given by the user to the media thing and the media thing controller (MController) that controls the media things. This paper introduces the contents of ISO/IEC 23093-5.

Keywords : Internet of media things, autonomous collaboration, MThing controller, holistic mission description, partial mission description

Ⅰ. Introduction

Internet of Media Things (IoMT) aims to provide intelligent media services to users by connecting (media) things to each other and exchanging media data. Media things (MThings) are composed of a media sensor (MSensor) that can acquire media and a media actuator (MActuator) that reproduces media. MThing also includes a media analyser (MAnalyser) that analyzes the obtained media and extracts meaningful information and a media storage (MStorage) that stores acquired media content or analyzed information. ISO/IEC 23093 Internet of Media Things standardizes data generated by media things and APIs that can use or exchange the generated data[1-3].

By connecting multiple media things, the mission given by the user is automatically executed, and intelligent media service can be provided to the user. The mission data defines the media thing that performs the task, the order in which the task is performed, and the specific type of task. ISO/IEC 23093-5 started standardization work on the mission data given by the user to the media thing and the media thing controller (MController) that controls the media

a) MyongJi University
‡ Corresponding Author : Sang-Kyun Kim
E-mail: goldmunt@gmail.com
Tel: +82-2-300-0637
ORCID: http://orcid.org/0000-0002-2359-8709

※ This work was supported by the Industrial Strategic technology development program (2021016000, International standard developments of task data format and API for autonomous collaboration between media things) funded by the Ministry of Trade, industry & Energy (MI, Korea).
· Manuscript October 31, 2022; Revised December 6, 2022; Accepted December 6, 2022.

things. This paper introduces the contents of ISO/IEC 23093-5[4].

The composition of the paper is as follows. Section 2 describes the mission execution process, including the IoMT architecture, mission composition and transmission process, and media controller. Section 3 introduces the mission data format defined in the standard, and Section 4 describes the mission data instance created using the mission data format, followed by the conclusion in Section 5.

II. IoMT autonoumous collaboration

1. Scope

The IoMT autonomous collaboration standard (i.e., ISO/IEC 23093-5) specifies data formats and APIs for the mission management and control between media things (MThings) and end-users/system managers. Specifically, the following interfaces, protocols and associated media-related information representations are under the scope of this document, as depicted in Figure 1:

- structured data formats (XML) representing the mission assigned by the user to the network of IoMT, for the data formats referred to by 1;
- structured data formats (XML) representing user commands to one or several MThings, possibly in a modified form (e.g., a subset of 1), regarding interface 1';
- structured data format (XML) to monitor the mission progression referred to by 4,
- APIs referred to by 4 to report or monitor the mission progression; and,
- APIs referred to by 1 and 1' to exchange the data for mission management and control.

2. Mission composition and transmission

This section describes the processes and examples of how the mission data are generated, transmitted, modified, and utilised to accomplish automatic collaborations among MThings.

Figure 2 exemplifies how the mission data is generated, transmitted, and finalized. The process of composing and propagating the task descriptions of MThings could be as follows:

- The user selects other MThings to complete the task desired by the user through their Graphic User Interface (GUI) applications;
- After completing the connection between MThings, the MThing notifies the user that the link has been established;

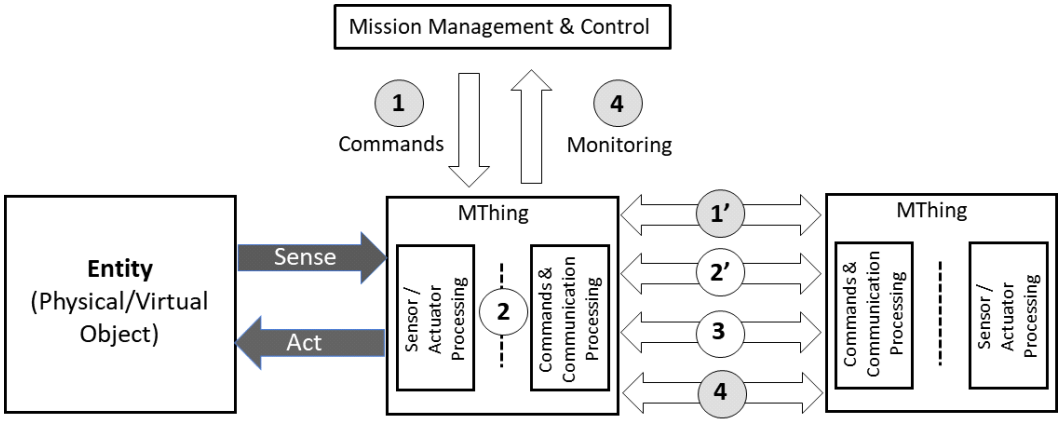

Fig. 1. Architectural view of the Internet of Media Things

- The user enters the role and execution order of each MThing. At this time, the user can describe or input the mission of each MThing through a graph tool in the GUI, menu button interfaces, or natural language,
- Converts the user's input into the mission data using the corresponding standardised data format (e.g., XML);
- The mission data (e.g., mission diagram) is presented to the user through the user's GUI to confirm it;
- The mission data verified by the user is modified for each participating MThing and broadcast to all connected MThings;
- The MThings start performing their respective tasks based on the transmitted mission data;
- Whenever each MThing completes its task recorded in the mission data, it broadcasts the completed status to other connected MThings (At this time, the user can check the task execution status of each MThing through their GUI);
- and finally, when all tasks in the mission data are completed, the last MThing delivers status information notifying the completion of the mission to connected MThings. The user can check the notification about the mission completion through their GUI.

This process exemplifies one of the practical implementations to show the mission data creation and propagation, therefore, not restricted to it.

3. Mission transition with MThing controller

The primary roles of the MThing controller (MController) for IoMT autonomous operation are as follows.

- Understand the abstract (e.g., holistic) level mission defined by the user through the application,
- Search for MThings required for a mission, connect available MThings,
- Create detailed (e.g., partial) level missions and deliver them to participating MThings;
- Initiate, pause, resume, or abort the mission;
- Monitor the status of participating MThings; and,
- Release the participating MThings after completing the mission.

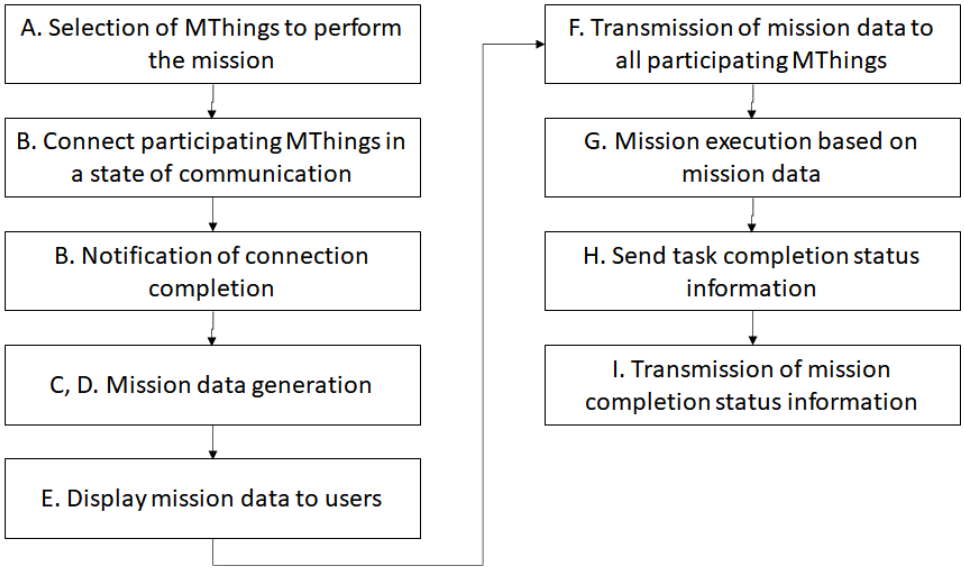

Fig. 2. Process of mission generation, transmission, and execution

Figure 3 shows a sequence diagram of a user and MThings, including the MController for autonomous operation. Suppose that a user wants to take a video from a nearby camera (i.e., MCamera), classify the captured vid-

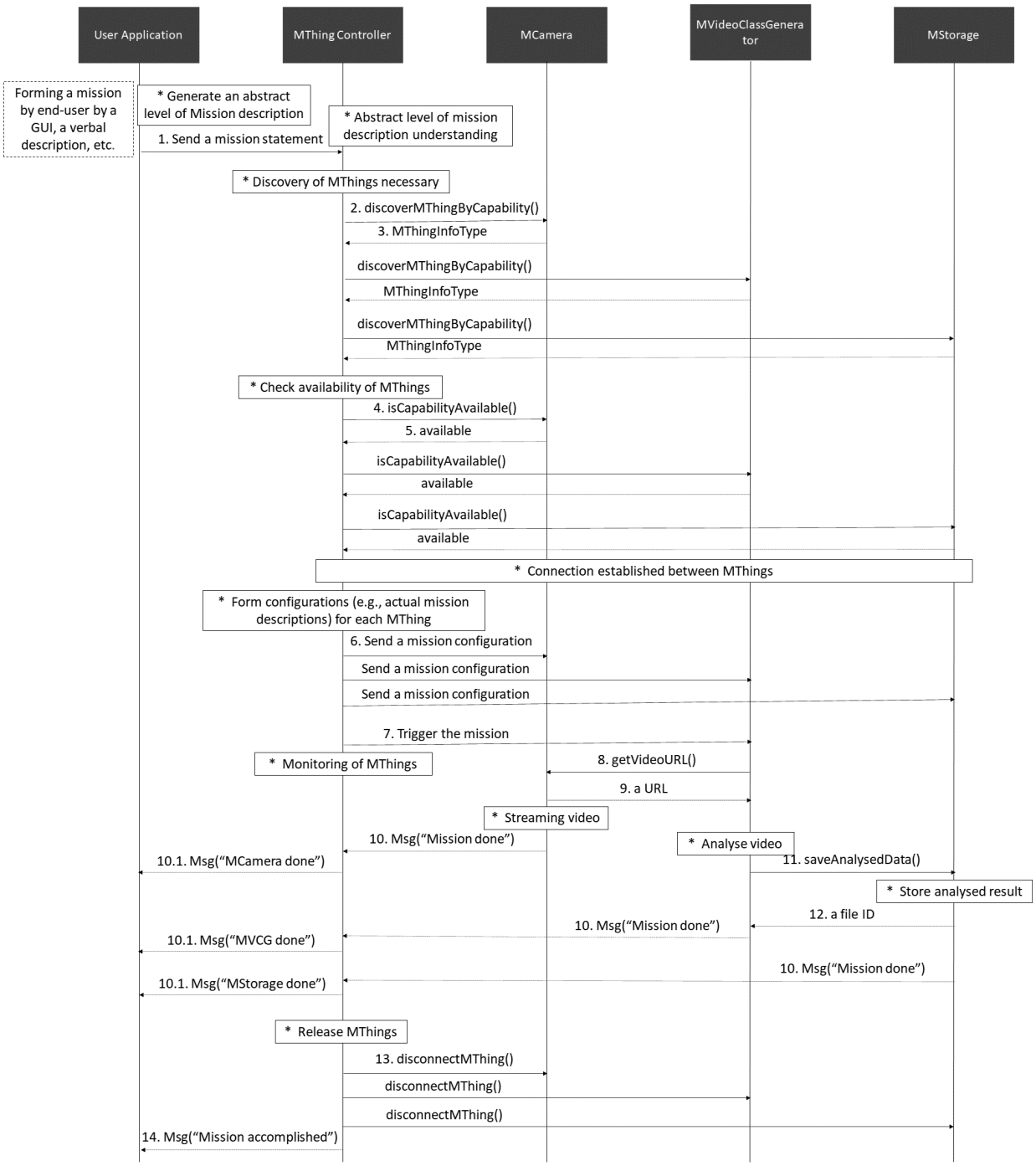

Fig. 3. Mission transition diagram

eo with an analyser (i.e., MVideoClassGenerator), and store the classified result in the storage (i.e., MStorage). The user can describe or input this task through a graph tool in the GUI, menu button interfaces, or natural language through the user applications.

The MController converts the task entered by the user into a holistic mission description (defined in Section 3.1) that conforms to the standard. Here, the holistic mission description includes the types and functions of MThings required to complete the task and how the functions are executed. The holistic mission description generated by the user application is sent to the MController for understanding (Figure 3-1). The MController searches the network for MThings required to complete the mission based on the understood mission using the function *discoverMThingByCapability()* (Figure 3-2) defined in ISO/IEC 23093-2[2]. The discovered MThings return an MThingInfoType (defined in ISO/IEC 23093-3[3]) containing their information to the MController (Figure 3-3). The MController asks whether the discovered MThing is available using the function *isCapabilityAvailable()* (Figure 3-4) and notifies the MController that the MThings are available (Figure 3-5).

The MController and other MThings are connected using the connection API defined in ISO/IEC 23093-2. After completing the connection between MThings, the MController creates a detailed task description (e.g., the partial mission description defined in Section 3.2) that each MThing must perform and sends the task description to all participating MThings. A partial mission description may include input and output information of each MThing, information on other MThings receiving output, and actions after task completion (Figure 3-6).

After the partial mission description is delivered to each MThing, the MController activates the mission of the MThing that starts the first mission (Figure 3-7). The MVideoClassGenerator checks its task and uses the function *getVideoURL()* (defined in ISO/IEC 23093-3) to request a URL for video streaming from MCamera (Figure 3-8). The MCamera returns the URL (Figure 3-9). Through the URL, the MVideoClassGenerator receives the streaming video as much as it requires. After streaming, the MCamera reports to the MController that its mission is complete (Figure 3-10). Upon receiving the mission completion message, the MController sends it back to the user application to notify them that the MCamera has completed the mission (Figure 3-10.1).

The MVideoClassGenerator, which has received video streaming from the MCamera, analyses the video and delivers the result to MStorage (Figure 3-11) using the function *saveAnalysedData()* (defined in ISO/IEC 23093-3). After saving the analysed result, the MStorage returns the file ID to the MVideoClassGenerator (Figure 3-12). The MStorage, which keeps the file, reports its mission completion to the MController, which in turn reports it to the user. After confirming that all tasks have been completed, the MController releases all connected MThings using the function *disconnectMThing()* (defined in ISO/IEC 23093-2) (Figure 3-13) and delivers the final task completion message to the user (Figure 3-14).

Again, this transition process exemplifies one of the practical implementations to show the mission carry-out, therefore, not restricted to it.

Ⅲ. Mission description data

1. Holistic mission data formats

The following are the semantics of elements and attributes that compose the holistic mission data format. The *MissionDescription* describes the mission's objectives, the MThings participating in it, and the order in which the mission is performed. The *MissionObjective* contains a

brief human-readable mission description. The *ParticipatingMThingList* describes the list of MThings participating in the mission. The *TriggerMThing* is the MThing that starts the very first stage of the mission. The *Orders* represent the order in which MThings are performed to achieve a mission. The *ParticipatingMThingListType* is a tool for de-

```
<element name="MissionDescription">
    <complexType>
        <sequence>
            <element name="MissionObjective" type="string" minOccurs="0" maxOccurs="1"/>
            <element name="ParticipatingMThingList" type="ParticipatingMThingListType"/>
            <element name="TriggerMThing" type="AdditionalMThingInfoType" minOccurs="0" maxOccurs="unbounded"/>
            <element name="Orders" type="OrderType"/>
        </sequence>
    </complexType>
</element>

<complexType name="ParticipatingMThingListType">
    <sequence minOccurs="1" maxOccurs="unbounded">
        <element name="ParticipatingMThing" type="ParticipatingMThingType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="ParticipatingMThingType">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <element name="MThingCapability">
            <complexType>
                <attribute name="requiredCapability" type="string" use="optional"/>
            </complexType>
        </element>
    </sequence>
    <attributeGroup ref="BasicMThingInfoBaseAttributes"/>
    <attribute name="idRef" type="string" use="optional"/>
</complexType>

<complexType name="AdditionalMThingInfoType">
    <attributeGroup ref="BasicMThingInfoBaseAttributes"/>
    <attribute name="idRef" type="string" use="optional"/>
    <attribute name="requiredCapability" type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="OrderType">
    <sequence>
        <element name="MThingFrom" minOccurs="0" maxOccurs="unbounded">
            <complexType>
                <sequence>
                    <element name="MThingTo" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                            <attributeGroup ref="BasicMThingInfoBaseAttributes"/>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </element>
    </sequence>
</complexType>

<attributeGroup name="BasicMThingInfoBaseAttributes">
    <attribute name="index" type="decimal" use="optional"/>
    <attribute name="mThing" type="mpeg7:termReferenceType" use="required"/>
</attributeGroup>
```

scribing the list of MThings participating in the mission. The *ParticipatingMThing* describes the MThings participating in the mission.

The *ParticipatingMThingType* is a tool for describing the MThing participating in the mission. The *MThingCapability* describes the capabilities that MThing can perform. The *requiredCapability* describes the capability of the corresponding MThing allocated to perform the mission.

The *AdditionalMThingInfoType* is a tool for describing MThing in more detail. The *idRef* is the unique identifier of MThing. The *requiredCapability* describes the capability of the corresponding MThing allocated to perform the mission.

The *OrderType* is a tool for describing the execution order in a step. The *MThingFrom* is an MThing that initiates an action. The *MThingTo* is an MThing(s) that finishes an action.

The *BasicMThingInfoBaseAttributes* is a tool for describing MThing in more detail. The *index* is an index to differentiate between MThings of the same type when participating in a mission. The *mThing* describes the kind of MThings.

2. Partial mission data formats

The following are the semantics of elements and attributes that compose the holistic mission data format. The *PartialMissionDescription* describes a partial mission that each MThing participating in the mission should perform. The *SetData* specifies to which MThing should give data. The *GetData* specifies from which MThing to request data. The *ConfirmCaller* confirms the MThing that invokes the function.

Ⅳ. Mission data examples

1. Execution sequence example

Figure 4 shows a sequence diagram demonstrating the request and response between multiple media things. In this example, an analyser requests two cameras to send capturing

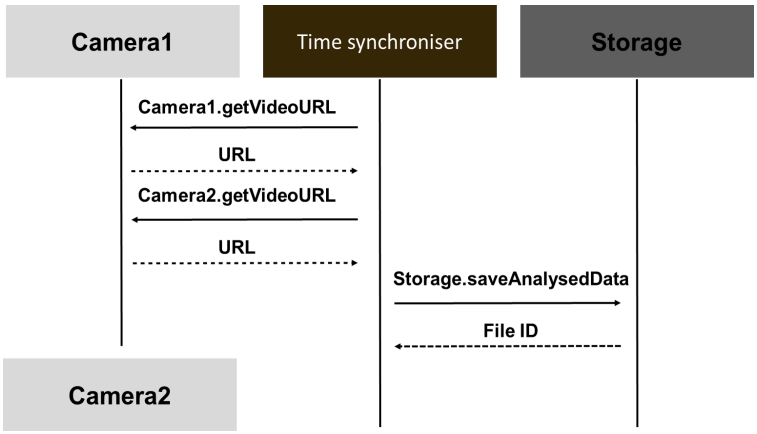

Fig. 4. Sequence diagram using multiple media things

```
<element name="PartialMissionDescription>
        <complexType>
                <element name="SetData" type="AdditionalMThingInfoType" minOccurs="0" maxOccurs=" unbounded "/>
                <element name="GetData" type="AdditionalMThingInfoType" minOccurs="0" maxOccurs="unbounded"/>
                <element name="ConfirmCaller" type="AdditionalMThingInfoType " minOccurs="0" maxOccurs="unbounded"/>
        </complexType>
</element>
```

videos, analyze the received videos to synchronise, and store the result in storage. The user application can form holistic mission data, and partial mission data generated by an MController can be sent to participating media things to carry out this mission autonomously, as in the following examples.

2. Holistic mission data example

In this example, an MTimeSynchroniser, two MCameras, and an MStorage participate in the mission. The mission starts with the MTimeSynchroniser, which sends commands to the two MCameras, and the MCameras return the requested ones to the MTimeSynchroniser. The MTimeSynchroniser then commands MStorage.

3. Partial mission data example

This example is a partial mission description received by MTimeSynchroniser from MController. MSynchroniser needs to send a "GetData" request to two MCameras, and a save request to MStorage.

```
<MissionDescription>
        <MissionObjective>
                Synchronise the time of the video taken by the two cameras and save it to the storage.
        </MissionObjective>
        <ParticipatingMThingList>
                <participatingMThing mThing="MTimeSynchroniser">
                        <MThingCapability requiredCapability="ANALYSER_SYNCHRONISE_TIME"/>
                </participatingMThing>
                <participatingMThing mThing="MCamera" index=1 idRef="CAM001">
                        <MThingCapability requiredCapability="SENSOR_CAPTURE_IMAGE"/>
                </participatingMThing>
                <participatingMThing mThing="MCamera" index=2 idRef="CAM002">
                        <MThingCapability requiredCapability="SENSOR_CAPTURE_IMAGE"/>
                </participatingMThing>
                <participatingMThing mThing="MStorage">
                        <MThingCapability requiredCapability="STORAGE_SAVE"/>
                </participatingMThing>
        </ParticipatingMThingsList>

        <TriggerMThing mThing="MTimeSynchroniser" requiredCapability="ANALYSER_SYCHRONISE_TIME"/>

        <Orders>
                <MThingFrom mThing="MTimeSynchroniser">
                        <MThingTo mThing="MCamera" index=1/>
                        <MThingTo mThing="MCamera" index=2/>
                </MThingFrom>
                <MThingFrom mThing="MCamera" index=1/>
                        <MThingTo mThing="MTimeSynchroniser"/>
                </MThingFrom>
                <MThingFrom mThing="MCamera" index=2/>
                        <MThingTo mThing="MTimeSynchroniser"/>
                </MThingFrom>
                <MThingFrom mThing="MTimeSynchroniser"/>
                        <MThingTo mThing="MStorage"/>
                </MThingFrom>
        </Orders>
</MissionDescription>
```

```
<PartialMissionDescription>
  <GetData mThingType="MCamera" idRef="CAM001" requiredCapability="SENSOR_CAPTURE_VIDEO"/>
  <GetData mThingType="MCamera" idRef="CAM002" requiredCapability="SENSOR_CAPTURE_VIDEO"/>
  <SetData mThingType="MStorage" idRef="STO001" requiredCapability="STORAGE_SAVE"/>
</PartialMissionDescription>
```

```
<PartialMissionDescription>
  <ConfirmCaller mThingType="MTimeSynchroniser" idRef="TSN001"/>
</PartialMissionDescription>
```

This example is a partial mission description that MCamera receives from MController. MCamera knows that the MThing it asks for is MTimeSynchroniser(TSN001).

Ⅴ. Conclusion

This paper briefly introduced the ISO/IEC 23093-5 Internet of media things – Autonomous collaboration standard currently being standardized. In the future, we plan to develop an API and data format that can exchange the task execution results of each media object. In addition, the standardization process for how to deal with the failure of media objects participating in the mission and how to rent media objects participating in sub-missions by multiple MControllers for large-scale mission performance will be included.

References

[1] “Text of ISO/IEC FDIS 23093-1 Ed 2 IoMT Architecture,” MDS20755_WG07_N00182, 135th MPEG, 2021.
[2] “Text of ISO/IEC FDIS 23093-2 Ed 2 IoMT Discovery and Communication API,” MDS20749_WG07_N00176, 135th MPEG, 2021.
[3] “Text of ISO/IEC FDIS 23093-3 Ed 2 IoMT Media Data Formats and API,” MDS20750_WG07_N00177, 135th MPEG, 2021.
[4] Sang-Kyun Kim, “WD 4 of ISO/IEC 23093-5 IoMT autonomous collaboration,” MDS21697_WG07_N00376, 139th MPEG, 2022.

Introduction Authors

Jonghoon Chun

- 1992 : Computer Science, Univ. of Denver, B.S.(1986), Northwestern Univ., M.S.(1988), Ph.D.(1992)
- 1992. 09. ~ 1995. 06. : Assistant Professor of Computing Science, Univ. of Central Oklahoma
- 1995. 09. ~ 2016. 02. : Professor of Computer Engineering, Myongji University
- 2016. 03. ~ Current : Professor of Data Technology, Myongji University
- 2011. 04. ~ Current : President and CEO, Prompt Technology Co., Ltd.
- ORCID : https://orcid.org/0000-0003-3396-4239
- Research interests : Database, Dataset Retrieval, Intelligence Software

Sang-Kyun Kim

- 1997. : Computer Science, Univ. of Iowa, B.S.(1991), M.S.(1995), PhD(1997)
- 1997. 03. ~ 2007. 02. : Professional Researcher, Multimedia Lab. of Samsung Advanced Institute of Technology
- 2007. 03. ~ 2016. 02. : Professor of Computer Engineering, Myongji University
- 2016. 03. ~ Current : Professor of Software Convergent, Myongji University
- ORCID : https://orcid.org/0000-0002-2359-8709
- Research interests : Digital Content(image, video and audio) analysis and management, 4D media, Blockchain, VR, Internet of Things and multimedia standardization