|  |
| --- |
| Input Contribution |
| Meeting ID\* | SDS#40 |
| Title:\* | TR-0043-Modbus-interworking-procedures |
| Source:\* | JaeSeung Song, KETI, jssong@sejong.ac.kr Sherzod Elamanov, KETI, selamanov@gmail.com  |
| Date:\* | 2019-05-13 |
| Input related to\* | TR-0043 Modbus Interworking |
| Intended purpose ofdocument:\* | [x]  Decision[ ]  Discussion[ ]  Information[ ]  Other <specify> |
| Impacted other TS/TR(s) | N/A |
| Decision requested or recommendation:\* | Add Section 7 Modbus interworking procedures |
| Template Version: January 2017 (Do not modify) |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

# Introduction

This contribution proposes to add a new section for Modbus Interworking procedures.

### -----------------------Start of change 1-------------------------------------------

# 7 Possible Solutions for oneM2M and Modbus Interworking

*This clause studies the possible solutions to realize oneM2M interworking with Modbus. Modbus-based devices can interwork with oneM2M system by usage of IPE that deploys on ASN, MN and IN, such as Modbus-based device connects to MN by IPE on MN. Resource mapping based on Modbus data model and operational procedure will be studied. Semantic method will also be consided in the solution.*

## 7.1 Exposure of Modbus Functions to the oneM2M System

### 7.1.1 Summary of Interworking Architecture for exposure of Modbus Functions

A Modbus-IPE exposing Modbus Functions to the oneM2M System is responsible for the creation of oneM2M Resources representing the exposed Modbus Functions on its own Registrar CSE. A single Modbus-IPE may expose Modbus Functions provided by one or more Modbus slave devices to the oneM2M System. A high-level summary of the relationship of Modbus devices providing Modbus Functions to be exposed to oneM2M and Modbus-IPE representing the exposed Modbus Functions is depicted in figure 7.1.1-1.



**Figure 7.1.1-1 Exposure of Modbus Functions to the oneM2M System**

## 7.1.2 Registration

#### 7.1.2.1 <*AE*> resource representing a Modbus-IPE

When a Modbus-IPE completes registration with its Registrar CSE, an <*AE*> resource representing that Modbus-IPE has been created as a result of that registration. This resource will be a parent resource of all Modubs devices connected to Modbus Master.

#### 7.1.2.2 <*flexContainer*> resource representing a Modbus device

Services provided by Modbus devices that are intended to be exposed to the oneM2M system shall be represented by <*flexContainer*> child resources of the Modbus-IPE <*AE*> resource instance in order to be exposed to the oneM2M system.

#### 7.1.2.3 <*container*> resource representing a Modbus device register group

Registers of Modbus device belong to one of 4 groups: read-only 1 byte, read&write 1 byte, read-only 16 byte, read&write 16 byte. Each register group should be exposed to the oneM2M system as a <*container*> resource.

#### 7.1.2.4 Sample oneM2M resource tree of Modbus-IPE

Figure 7.1.2.4-1 shows sample oneM2M resource structure for Modbus interworking. IPE\_AE is *<AE>* resource which is a parent of all Modbus devices. Device\_1\_flex\_cnt is a *<flexContainer>* resource representing a Modbus-IPE. The *<flexContainer>* resources are named according to device ids. Device\_1\_flex\_cnt has 4 children <*container*> resources representing 4 groups of internal registers: coil\_r\_cnt (read-only 1 byte register), coil\_rw\_cnt (read&write 1 byte register), register\_r\_cnt (read-only 16 byte register), register\_rw\_cnt (read&write 16 byte register). <*Subscription*> resources are required for monitoring.



**Figure 7.1.2.4-1 oneM2M resource structure for Modbus interworking**

#### 7.1.2.5 Modbus device registration call flow

Figure 7.1.2.5-1 shows the device registration call flow.

1. Modbus IPE sends Create *<AE>* request to CSE to register Modbus Master.
2. Modbus devices are registered at Modbus Master.
3. Modbus IPE sends corresponding requests to CSE to create a resourse structure as in Figure 7.1.2.4-1.

#

**Figure 7.1.2.5-1 Device registration call flow**

## 7.1.3 Monitoring

#### 7.1.3.1 Procedure for monitoring Modbus device

The requests order for retrieiving data from Modbus device is showed in Figure 7.1.3.2-1. For continuous monitoring the procedure should repeat continiuosly.

1. Modbus-IPE sends retrieve request to Modbus device.

2. Modbus device response with measured data to Modbus-IPE.

3. IPE sends create *<contentInstance>* request resource to Modbus-CSE.

4. CSE sends create <*contentInstance*> response to Modbus-IPE.

5. As device subscribed to <*container*> resource it is notified with measured data from sensor.



**Figure 7.1.3.1-1 Flow diagram for monitoring Modbus device**

#### 7.1.3.2 *<contentInstance>* resource for device monitoring

Modbus-IPE uses Create <*contentInstance*> request to send data received from Modbus devices. When Modbus-IPE sends Create <*contentInstance*> request, it should specify the corresponding <*flexContainer*> and <*container*> resources IDs representing Modbus device and register group resprectively. The <*contentInstance*> should correspond to common format which would conform the Modbus device structure. The common <*contentInstance*> format will be described further.

Storage of Modbus devices are register based. Each register has its own logical address which should be used to query a register from Modbus IPE. In some Modbus devices apart from logical addresses other register charachtersitcs exist, for example, read/write, PDU address, data size, default value. Considering above facts, the mandatory attributes that should be specified to create <*contentInstance*> are logical address of a register and its value. Other attributes are optional. For example, the minimal body of Create *<contentInstance>* request can have the following format:

{
    **"m2m:cin"**:{
        **"con"**:{
            **"address"**:"40011",
            **"value"**: 22
        }
    }
}

### -----------------------End of change 1 -------------------------------------------