|  |
| --- |
| 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 Registration

Figure 7.1-1 shows oneM2M resource structure for Modbus interworking. IPE\_AE is AE resource which is a parent of all Modbus devices. Each device is projected as a container resource under IPE\_AE resource. The additional address of Modbus RTU frame (1 byte) represents unique device id. So, the container resources are named according to their id. Each device container has 4 child containers 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 and pollingChannel resources are required for monitoring (described in next section).



**Figure 7.1-1 oneM2M resource structure for Modbus interworking**

Figure 7.1-2 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 7.2

#

**Figure 7.1-2 Device registration call flow**

## 7.2 Monitoring

To understand the nature of Modbus devices let’s examine real Modbus device. The Table 7.2-1 represents address table of temperature sensor tst399v2 working on Modbus. We can see that each register has several characteristics (register name, r/w, fc etc). So it is reasonable to write content instances as JSON (or XML) objects. For example:

{

 rw: true,

 fc: “03/06”

 pdu\_address: 100,

 offset: 40001,

 datasize: 32,

 value: 22

}

By looking at pdu\_address and offset, we can understand the register represents measured temperature. The measured value is 22.



**Table 7.2-1 Temperature sensor address table**

As Modbus devices does not have notification mechanism, polling should be used for continuous monitoring. That’s why pollingChannel resource is created as child resource of IPE\_AE for this purpose (Refer to Figure 7.1-1).

1. IPE sends retrieve requests to Modbus device periodically.

2. Modbus device responses with measured data to IPE.

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

4. CSE sends create < contentInstance > response to IPE

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



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