

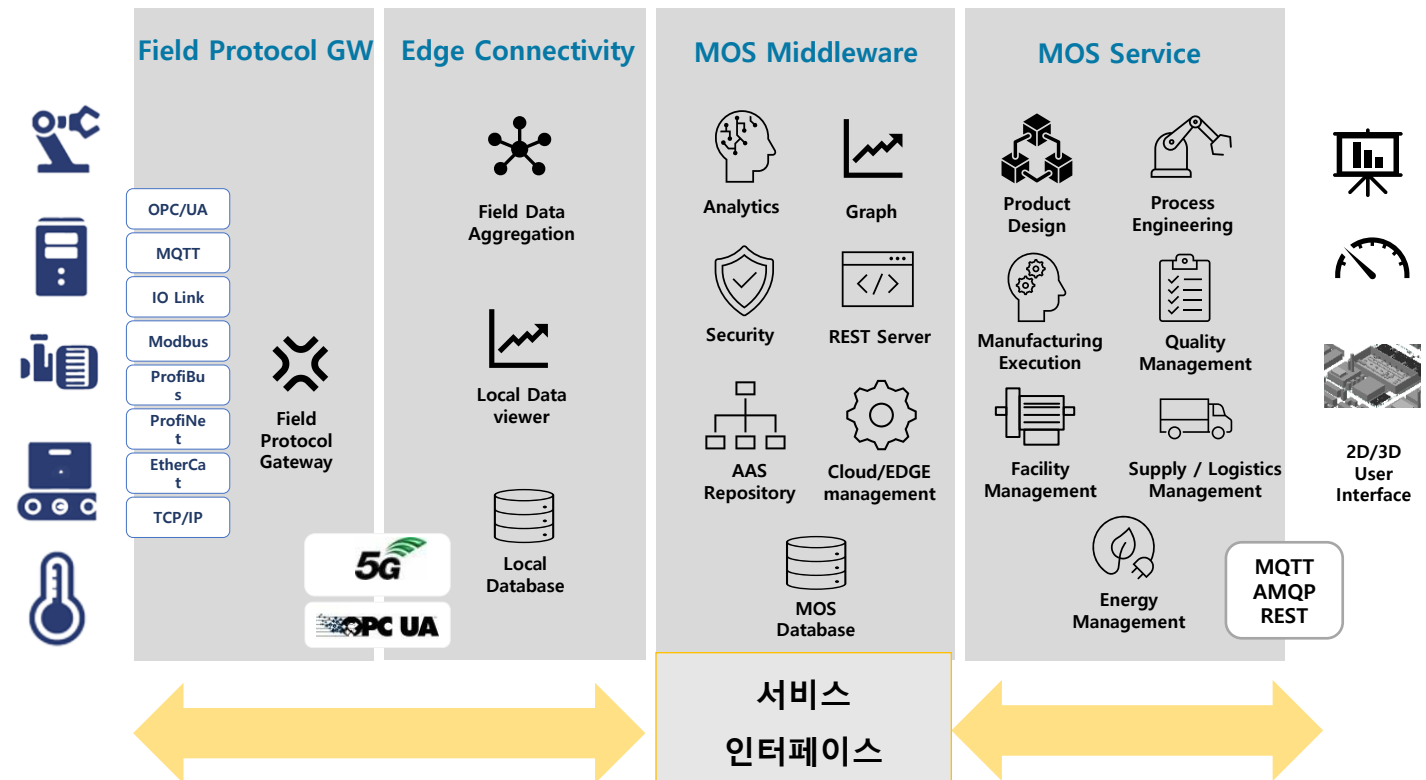
# 제조운영체제 서비스 인터페이스 규격

## MOS Service Interface Specification



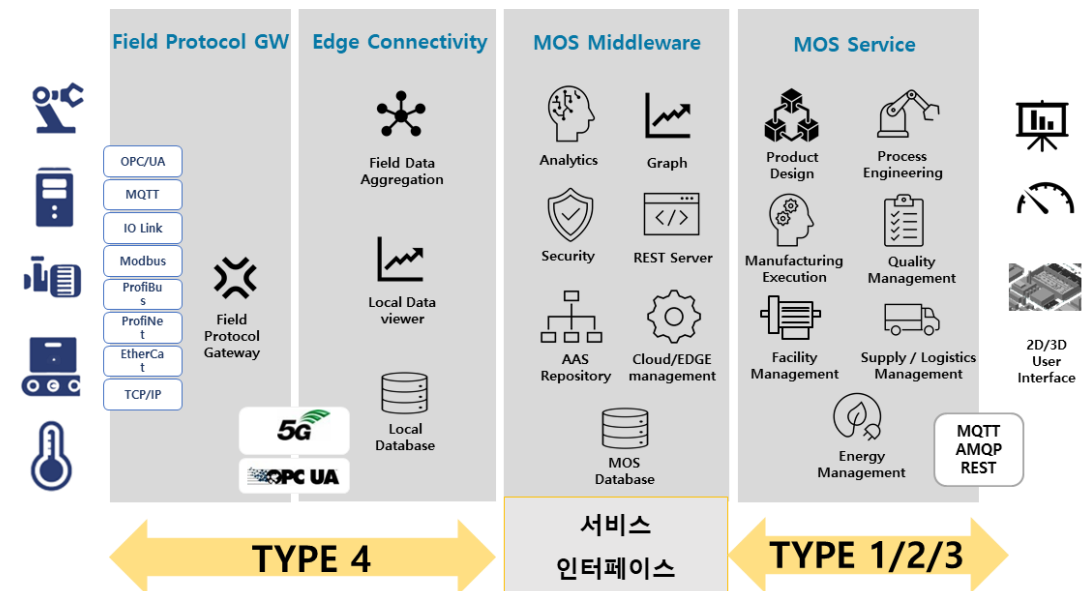
2023-11-06

- 서비스 인터페이스 규격이란?
  - AAS 기반 제조운영체제와 외부 어플리케이션 서비스와의 데이터 인터페이스 방법을 정의
  - 제조 현장의 IOT 필드장비와 제조운영체제와의 직접적인 데이터 인터페이스 방법을 정의



- 서비스 인터페이스 타입

- **TYPE 1** AMQP 인터페이스 – 실시간으로 수집되는 데이터를 수신하기 위한 인터페이스
- **TYPE 2** REST Server 인터페이스 – 저장된 데이터에 접근하거나 값을 변경하기 위한 인터페이스
- **TYPE 3** REST Client 인터페이스 – 외부 서비스로부터 데이터를 가져오기 위한 인터페이스
- **TYPE 4** IoT MQTT 인터페이스 – MQTT를 사용하여 필드 데이터를 직접 수집하는 인터페이스





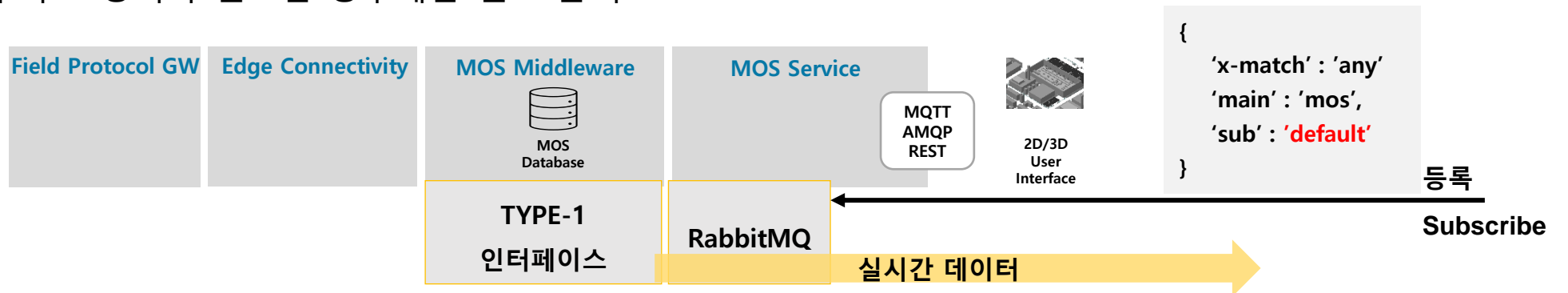
**Smart  
Manufacturing**

**Digital  
Twin**

**Asset  
Administration  
Shell**

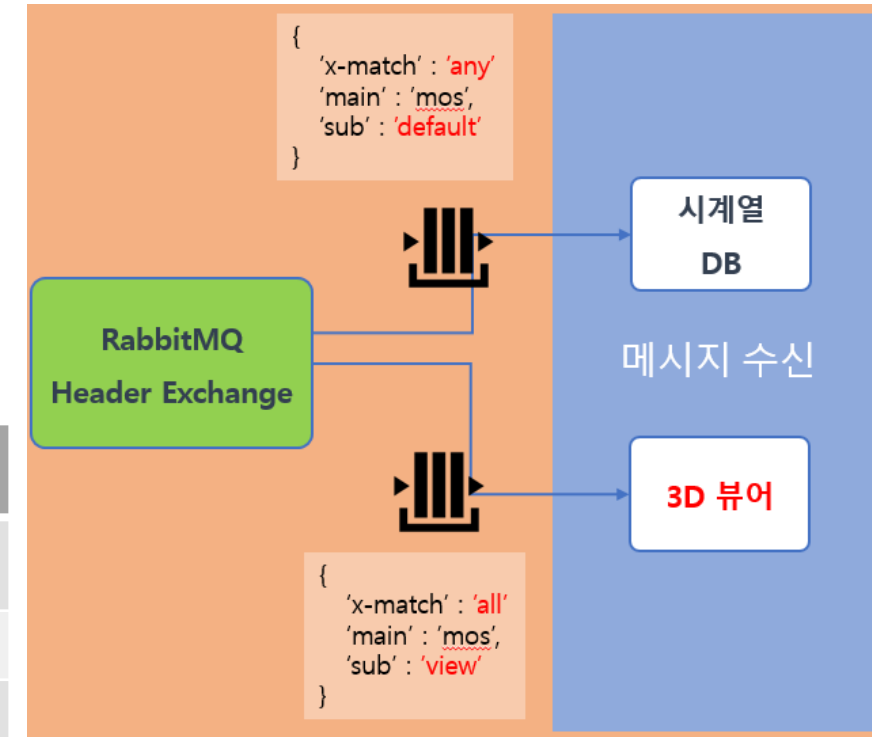
**TYPE 1. AMQP 인터페이스 (실시간 데이터)**

- RabbitMQ Header Exchange 방식의 인터페이스
  - 사용목적 – 제조운영체제가 수집하는 데이터를 동시에 수신
  - 사용방법 – 외부서비스에서 실시간 데이터 수신을 등록 (Subscribe)
  - 접속방법 – 제조운영체제의 AMQP 접속→등록→데이터수신 (접속 주소는 현장별로 다름)
  - 등록방법 – 수신할 데이터 종류를 『sub』항목에 지정
    - 'default' – 수집되는 모든 데이터를 수신
    - 'view' – 수집되는 데이터 중 3D 대시보드에서 표출하도록 설정된 데이터만 수신
    - 이 외에 추가로 정의가 필요한 경우에는 별도 문의



- 『x-match』 파라미터 설정
  - 'any' – 헤더 테이블 값이 하나라도 일치하면 수신
  - 'all' 헤더 테이블 값이 모두 일치하면 수신

Header		
'main'	'mos'	제조운영체제 데이터 태그
'sub'	'default'	수집되는 모든 데이터를 수신
	'view'	3D Dashboard에 표시하도록 설정된 데이터만을 수신



- 제조운영체제가 전송(Publish)하는 데이터 형식 (JSON)

'time'	데이터가 수집된 timestamp (1970년 이후 경과된 시간, nano-seconds)
'tag'	수집된 데이터 TAG NAME
'value'	데이터의 실제 값

```
time: 1632291999815269000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_Z, value: 710.075928
time: 1632291999815274000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_U, value: 90.195099
time: 1632291999815280000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_V, value: -0.612665
time: 1632291999815285000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_W, value: -179.462509
time: 1632292000027534000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_X, value: 28.324600
time: 1632292000027562000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_Y, value: 483.080200
time: 1632292000027570000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_Z, value: 710.063232
time: 1632292000027577000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_U, value: 90.188995
time: 1632292000027584000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_V, value: -0.567665
time: 1632292000027590000, tag: ns=2;s=ProSix6A106F156S.OperationalData.CurrentPosition.World.Position_W, value: -179.437210
```

# 클라이언트 예제 코드 (Python)

```
1  #!/usr/bin/env python3
2  import json
3  import pika
4  import sys
5
6  exchangeName = 'mos_exchanges'
7  exchangeType = 'headers'
8
9  count = 0
10
11  # callback
12  def callback(channel, method, properties, body):
13      # print("Channel %r" % channel)
14      # print("Method %r" % method)
15      # print("Properties %r" % properties)
16      # print("Body %r" % body)
17      # print(body)
18      data = json.loads(body)
19      print("time: %s, tag: %s, value: %s" % (data['time'], data['field'], data['value']))
20
21  # connection
22  credentials = pika.PlainCredentials(username='guest', password='guest')
23  connection = pika.BlockingConnection(pika.ConnectionParameters(host='192.168.0.42', port=30400, credentials=credentials))
24  channel = connection.channel()
25
26  # queue
27  channel.exchange_declare(exchange=exchangeName, exchange_type=exchangeType)
28  result = channel.queue_declare('', exclusive=True)
29  queueName = result.method.queue
30
31  # binding
32  headers = {}
33  headers["x-match"] = 'all'
34  headers["main-app"] = 'mos'
35  headers["sub-app"] = 'view'
36  channel.queue_bind(
37      exchange=exchangeName,
38      queue=queueName,
39      arguments=headers)
40
41  # consume
42  print("Waiting for messages. To exit press CTRL+C")
43  channel.basic_consume(queue=queueName, on_message_callback=callback, auto_ack=True)
44  channel.start_consuming()
```





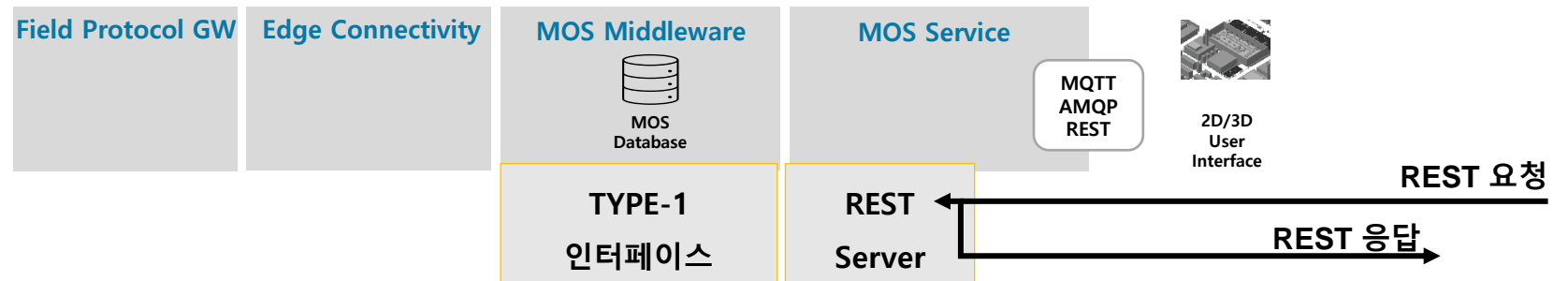
**Smart  
Manufacturing**

**Digital  
Twin**

**Asset  
Administration  
Shell**

**TYPE 2. REST Server 인터페이스 (저장 데이터 접근 및 변경)**

- 제조운영체제에 저장된 AAS 기반 데이터를 조회할 수 있는 REST Server
  - 사용목적
    - 외부 서비스에서 제조운영체제가 수집/저장한 이력 데이터 조회
    - 외부 서비스에서 제조운영체제가 마지막으로 수집/저장한 데이터 조회
  - 사용방법 – 외부 서비스가 REST Client 역할을 하며, 제조운영체제에 저장된 데이터 요청



- 용도 – 제조운영체제에 저장된 AAS 태그 리스트를 요청 (InfluxDB의 mos 버킷)
- 서비스 – HTTP/GET
- 서비스 URL - /api/meta
- 클라이언트 인증 – API KEY 방식 (Request 헤더에 포함하여 전송)

## (예시)

- URL : `http://[MOS Cloud IP]:7770/api/meta`
- Header : `Authorization: Api-Key CLZ86OVKUYIPO7XLDGFNVJLXGDGNBVZUZDFYST36ZR87PEC57L`

# AAS 태그 리스트 조회 예시

태그 리스트 요청 (외부 서비스) – CURL 사용 예	태그 리스트 응답 (제조운영체제)
<pre>curl -X GET http://127.0.0.1:7770/api/<b>meta</b> -H "Authorization: Api-Key Uoxgypyus3fdDaawqq7wejdopcA2waW3QmsalweprT0_3pPaqw2=="</pre>	<p>- 조회 성공 시</p> <pre>{   "code": 1,   "message": "Success",   "data":   [     { "NAME": "ns=2;s=Productions.Machine_1.Operational_Data.Annual_Result"},     { "NAME": "ns=2;s=Productions.Machine_1.Operational_Data. Daily_Result"},     ..... 동일한 구조가 반복됨 .....   ] }</pre>
	<p>- 조회 실패 시</p> <pre>{   "code" : 0,   "message": "Failed",   "details": "[오류 메시지]" }</pre>

- 용도 - 제조운영체제에 저장된 『지정된 기간 동안 저장된 AAS 태그 데이터』 요청
- 서비스 - HTTP/GET
- 서비스 URL - /api/tag
- 클라이언트 인증 - API KEY 방식 (Request 헤더에 포함하여 전송)

파라미터	상세 내용
"tag"	조회하고자 하는 태그 이름 (URL Encode 후 사용)
"from"	조회하고자 하는 시간구간의 시작시점. (포맷 : %Y%m%d%H%M%S, ex : 20220817114729 <- 2022년 8월 17일 11시 47분 29초)
"to"	조회하고자 하는 시간구간의 종료시점. (포맷 : %Y%m%d%H%M%S, ex : 20220817114729 <- 2022년 8월 17일 11시 47분 29초)
"last"	최신 값 조회 여부 (설정 시 'from', 'to' 파라미터는 무시함 )

# AAS 태그 데이터 조회 예시

## 태그 데이터 요청 (외부 서비스) – CURL 사용 예

- 기간을 지정해서 조회  
curl -X GET  
http://127.0.0.1:7770/api/**tag**?tag=ns%3D%3Bs%3DRobot.OperationalData.Status&**from**=2023101900000000&**to**=2023103000000000 -H  
"Authorization: Api-Key  
Uoxgypyus3fdDaawqq7wejdopcA2waW3QmsalweprT0\_3pPaqw2=="

※ UTC 기준으로, 한국 시간보다 9시간 느린 시점으로 설정  
(ex. 한국 시각 14:00 데이터 조회 시 05:00 로 설정)

## 태그 데이터 응답 (제조운영체제)

- 조회 성공 시 응답

```
{
  "code" : 1,
  "message": "Success",
  "data": [
    { "NAME": "~~~", "TIME": "2023-10-20T:09:18:43.846195Z", "VALUE": "abc" },
    { "NAME": "~~~", "TIME": "2023-10-20T:09:18:45.532144Z", "VALUE": "def" },
    ..동일 구조 반복..
  ]
}
```

※ UTC 기준 시각으로 한국 표준시와 9시간 차이

- 조회 실패 시 응답

```
{
  "code" : 0,
  "message": "Failed" or "No data"
  "details": "[오류 메시지]"
}
```

# AAS 태그 데이터 조회 예시

## 태그 데이터 요청 (외부 서비스) – CURL 사용 예

- 최종 데이터 하나만 조회  
curl -X GET  
http://127.0.0.1:7770/api/**tag**?tag=ns%3D%3Bs%3DRobot.OperationalData.Status&**last** -H "Authorization: Api-Key  
Uoxgypyus3fdDaawqq7wejdopcA2waW3QmsalweprT0\_3pPaqw2=="

## 태그 데이터 응답 (제조운영체제)

- 조회 성공 시 응답  
{  
 "code" : 1,  
 "message": "Success",  
 "data": [  
 { "NAME": "~~~", "TIME": "2023-10-20T:09:18:43.846195Z", "VALUE": "abc" }  
 ]  
}

※ UTC 기준 시각으로 한국 표준시와 9시간 차이

- 조회 실패 시 응답  
{  
 "code" : 0,  
 "message": "Failed" or "No data"  
 "details": "[오류 메시지]"  
}

- 용도 - 제조운영체제에 등록된 하나의 『AAS 태그 데이터』 현재 값을 변경

※ 제조운영체제에서 『변경 가능하도록 설정한 AAS 태그』에 대해서만 현재 값 변경 가능

- 서비스 - HTTP/POST 또는 PUT
- 서비스 URL - /api/tag

파라미터		상세 내용
"tag"	조회하고자 하는 태그 이름	
"value"	변경하고자 하는 값	



# AAS 태그 데이터 변경 예시

## 태그 데이터 요청 (외부 서비스) – CURL 사용 예

```
curl -X POST http://127.0.0.1:7770/api/tag --data  
"@put.json" -H "Content-Type: application/json"
```

### [body 데이터 형식 (JSON)]

```
{  
  "tag": "ns=2;s=Productions.Machine_8.Actual_Position",  
  "value": 1  
}
```

## 태그 데이터 응답 (제조운영체제)

### - 조회 성공 시 응답

```
{  
  "code": 1,  
  "message": "Success"  
}
```

### - 조회 실패 시 응답

```
{  
  "code": 0,  
  "message": "Failed",  
  "details": "[오류 메시지]"  
}
```

```
{  
  "tag": "abcdefg",  
  "value": 123456  
}
```

- 용도 – 타 저장소에 저장된 제조데이터를 블록단위로 AAS 시계열 데이터베이스에 업로드
- 서비스 – HTTP/POST 또는 PUT
- 서비스 URL - /api/tag/block

파라미터		상세 내용
"block"	데이터가 담긴 배열 블록	
"tag"	조회하고자 하는 태그 이름	
"value"	변경하고자 하는 값	
"timestamp"	각 태그가 저장될 시점(nanosecond단위의 unix timestamp)	

# AAS 블록 태그 데이터 업로드 예시

## 태그 데이터 요청 (외부 서비스) – CURL 사용 예

```
curl -X POST http://127.0.0.1:7770/api/tag/block --data  
"@put.json" -H "Content-Type: application/json"
```

### [body 데이터 형식 (JSON)]

```
{  
  "block": [  
    { "tag": " ns=2;s=Productions.Machine_8.Actual_Position", "value": 123,  
      "timestamp": 1699211523000000000 },  
    { "tag": " ns=2;s=Productions.Machine_8.Actual_Temp", "value": 456,  
      "timestamp": 1699211524000000000 }  
  ]  
}
```

## 태그 데이터 응답 (제조운영체제)

### - 조회 성공 시 응답

```
{  
  "code": 1,  
  "message": "Success"  
}
```

### - 조회 실패 시 응답

```
{  
  "code": 0,  
  "message": "Failed",  
  "details": "[오류 메시지]"  
}
```

```
{  
  "block": [  
    { "tag": "aasTag1", "value": 123, "timestamp": 1699323972000000000 },  
    { "tag": "aasTag2", "value": 456, "timestamp": 1699323973000000000 }  
  ]  
}
```



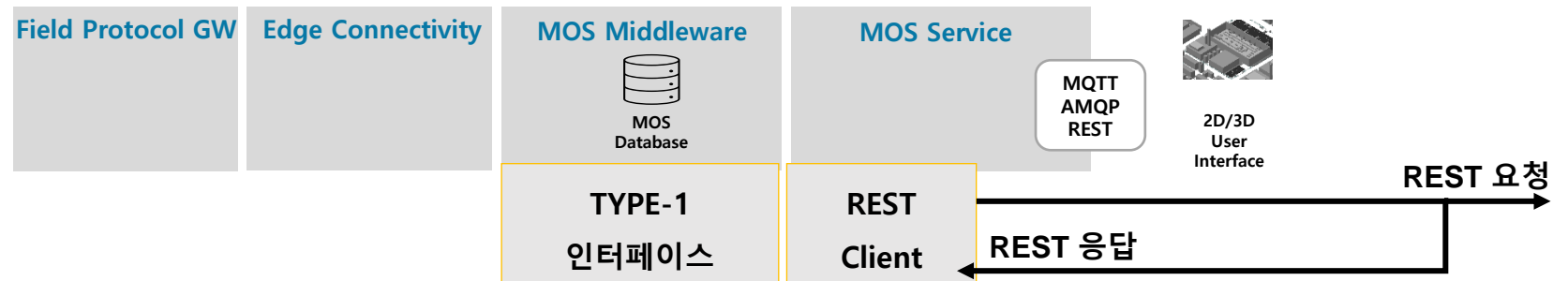
**Smart  
Manufacturing**

**Digital  
Twin**

**Asset  
Administration  
Shell**

**TYPE 3. REST Client 인터페이스 (외부데이터 업데이트)**

- 제조운영체제에서 외부 서비스로부터 주기적으로 데이터를 가져오는 REST Client
  - 사용목적
    - MES/ERP와 같은 관리용 서비스로부터 주요 지표 데이터를 제조운영체제로 입력하는 경우
    - 제조운영체제에서 주기적으로 대량의 데이터를 업데이트 하는 경우
  - 사용방법 – 제조운영체제가 REST Client 역할을 하며, 외부 서비스에 저장된 데이터 요청
    - ※ 데이터는 미리 AAS에 모델링된 AAS 태그명으로 식별됨



- 용도 - 제조운영체제에 『사전에 상호 합의된 데이터 값 리스트』를 주기적으로 요청
- 서비스 - HTTPS 또는 HTTP /GET (외부 서비스가 제공하는 REST Server가 지원하는 방식에 따름)
- 서비스 URL - /api/meta
- 클라이언트 인증 - API KEY 방식 (Request 헤더에 포함하여 전송)
- 요청하는 URL과 그에 따른 응답 데이터 리스트는 제조운영체제와 외부서비스가 사전 협의해야 함

(예시)

**GET** /product/ HTTP/1.1

Host: 192.168.110.33:9440

**Authorization: Api-Key** hGSiLw9Q.EvuhbWEXKCYAvfXPhz5dhKx2p97ACLPI

- 용도 – 제조운영체제로 『사전에 상호 합의된 데이터 값 리스트』를 JSON 형태로 제공
- 응답은 BODY에 JSON 형식으로 인코딩하여 전달
- 개별 데이터마다 값이 변경된 Timestamp 함께 제공 (제조운영체제는 Timestamp를 함께 저장)
- 데이터 항목 하나당 하나의 JSON 데이터가 인코딩 되며, 아래와 같은 형식으로 인코딩 됨

```
{ "tag" : "CA03.RUNST.ST",    "value" : "1",    "time" : 1632291999815269000 }
```

파라미터	상세 내용
"tag"	AAS 모델에 정의된 태그명
"value"	실제 값. 데이터 타입은 AAS에 명시되어 있으며, 제조운영체제에서 데이터 타입에 따라 파싱됨
"time"	데이터가 변경된 타임스탬프 (1970년 이후 경과된 시간을 nano-seconds로 표현한 값)

※ "value"에 대한 data type은 매칭될 AAS Property에 설정되어 있어 JSON 파라미터에는 포함하지 않음

```
[  
  {  
    "module" : "Product",  
    "data" : [  
      { "tag" : "Product.CA03.Operation.RunState",      "value" : "RUN",    "time" : 1632291999815269000 },  
      { "tag" : "Product.CA03.Production.ProductionQty",  "value" : "1",      "time" : 1632291999815269000 },  
      .....  
    ]  
  }  
]
```

실제 교환되는 데이터 값 배열 (제조운영체제 ← 외부서비스)





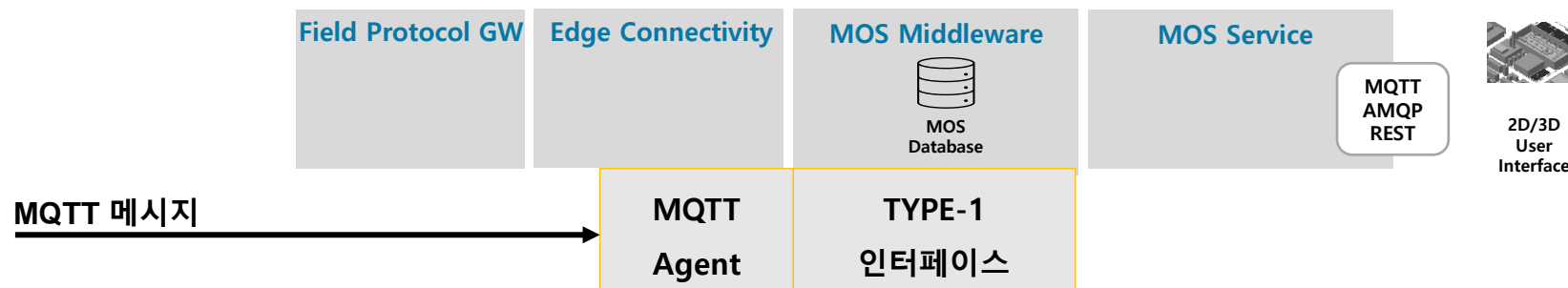
**Smart  
Manufacturing**

**Digital  
Twin**

**Asset  
Administration  
Shell**

**TYPE 4. IoT MQTT 인터페이스 (필드데이터 직접 수집)**

- 제조운영체제에서 외부 서비스로부터 주기적으로 데이터를 가져오는 REST Client
  - 사용목적
    - 제조 현장의 필드 디바이스가 제조운영체제에 직접 데이터를 전송
    - 단순한 구조와 제한된 리소스를 사용하는 센서/인터페이스 디바이스로부터의 데이터 수집
  - 사용방법 – 필드 디바이스는 주기적/비주기적으로 제조운영체제에 데이터 전송 (MQTT 사용)
    - ※ 데이터는 미리 AAS에 모델링된 AAS 태그명으로 식별됨
    - ※ 데이터는 JSON 형식으로 엔코딩되어 전송됨



- 용도 – 제조 현장의 필드 디바이스가 제조운영체제에 데이터를 전송
- MQTT 토픽 – mos/type-4/{디바이스 식별정보}
- MQTT 데이터 – 데이터 항목 하나당 다음과 같은 JSON 형식으로 인코딩되어 전송 (다음 페이지 참고)

```
{ "tag" : "CA03.RUNST.ST",   "value" : "1" , "timestamp": 1699822136000000000(Optional) }
```

- localtime와 timestamp는 nanosecond단위의 unix timestamp 사용
- 서비스 등록 및 실행은 아래 링크 참고

- <https://github.com/auto-mos/MOS-Packages/tree/main/Tiny%20Package/MOS%20Cloud/type4>

파라미터	상세 내용
"tag"	AAS 모델에 정의된 태그명. 태그명은 현장별로 작성된 AAS에 따라 다름
"value"	실제 값. 데이터 타입은 AAS에 명시되어 있으며, 제조운영체제에서 데이터 타입에 따라 파싱됨
"localtime"	디바이스에서 일괄적으로 데이터를 전송하는 시점으로, 0 이외의 값 입력 시 모든 태그 데이터에 적용됨
"timestamp"	태그데이터의 시간값을 개별적으로 전송하기 위해 사용하는 값

※ "value"에 대한 data type은 매칭될 AAS Property에 설정되어 있어 JSON 파라미터에는 포함하지 않음

## 1. 데이터 Timestamp 일괄 전송 메시지 예

MQTT 토픽 – mos/type-4/device-01

MQTT 데이터

```
{  
  "localtime" : "1632291999815269000",
```

필드 디바이스에서 전송 시점에 측정한 timestamp  
(1970년 이후 경과된 시간을 nano-seconds로 표현한 값)  
"0" 값 사용 시 각 태그에 지정된 timestamp 값을 사용하여 저장

```
  "data" : [  
    { "tag" : "Product.CA03.Operation.RunState", "value" : "RUN" },  
    { "tag" : "Product.CA03.Production.ProductionQty", "value" : "1" }  
    .....  
  ]  
}
```

실제 전달되는 데이터 값 배열 (필드 디바이스 → 제조운영체제)

## 2. 데이터 Timestamp 개별 전송 메시지 예

MQTT 토픽 – mos/type-4/device-01

MQTT 데이터

```
{  
  "localtime" : "0",
```

필드 디바이스에서 전송 시점에 측정한 timestamp  
(1970년 이후 경과된 시간을 nano-seconds로 표현한 값)  
"0" 값 사용 시 각 태그에 지정된 timestamp 값을 사용하여 저장

```
  "data" : [  
    { "tag" : "Product.CA03.Operation.RunState", "value" : "RUN", "timestamp":1698986710000000000 },  
    { "tag" : "Product.CA03.Production.ProductionQty", "value" : "1", "timestamp":1698986720000000000 },  
    .....  
  ]
```

실제 전달되는 데이터 값 배열 (필드 디바이스 → 제조운영체제)

```
}
```



**Smart  
Manufacturing**

**Digital  
Twin**

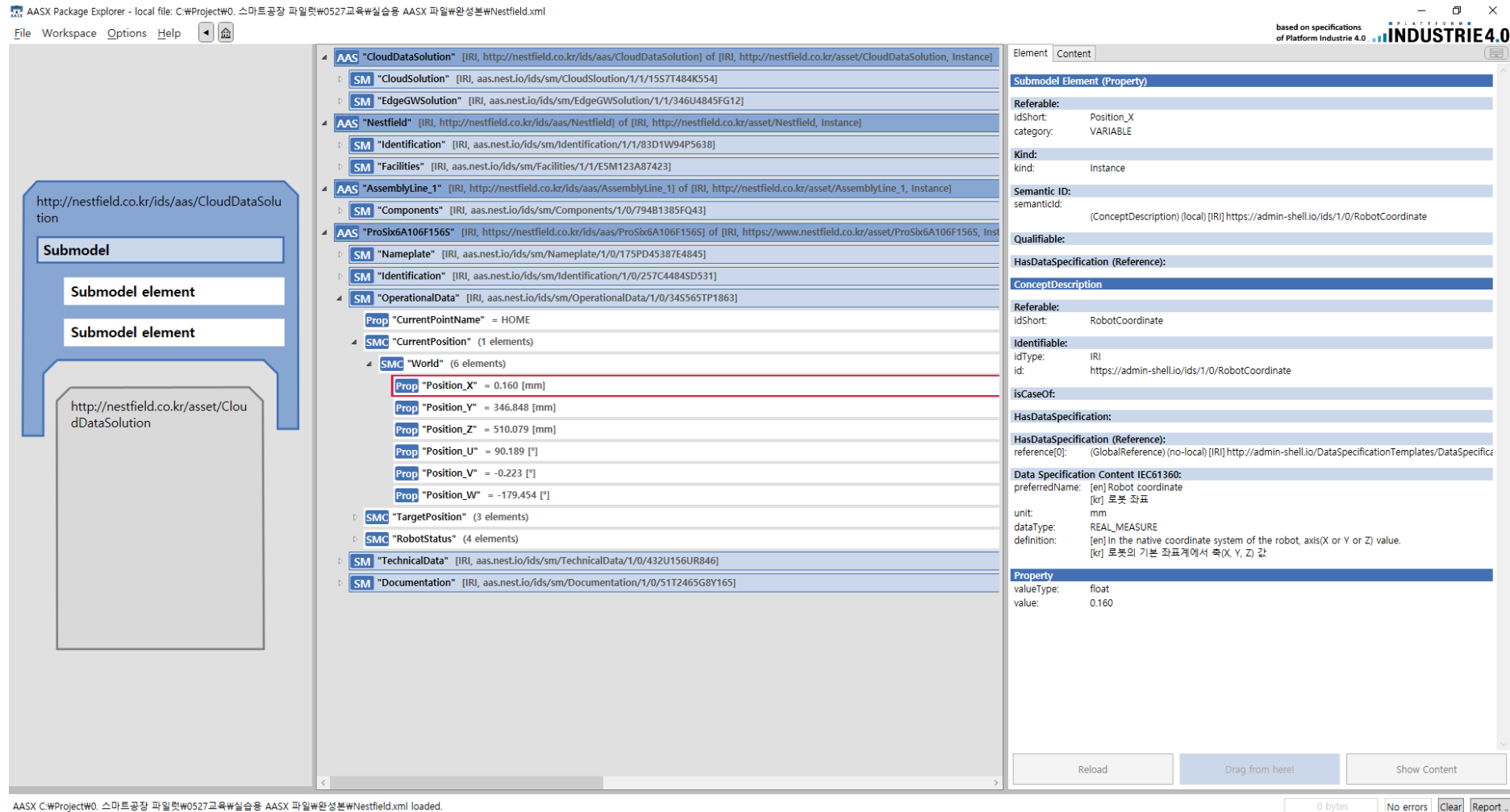
**Asset  
Administration  
Shell**

**AAS 태그명 확인 방법**



# 작성된 AAS를 MOS에 적용하는 절차

- 1단계 – Aasx Package Explorer를 사용한 AAS 작성



The screenshot displays the AASX Package Explorer interface, which is used for managing AAS (Autonomous Agent Specification) packages. The interface is divided into several panes:

- Left Pane:** Shows the package structure. The selected package is "CloudDataSolution" (IRI: http://nestfield.co.kr/ids/aas/CloudDataSolution). It contains submodels like "CloudSolution", "EdgeGWSolution", "Nestfield", "Identification", "Facilities", "AssemblyLine\_1", "ProSix6A106F1565", "Nameplate", "OperationalData", "CurrentPosition", "World", "TargetPosition", "RobotStatus", "TechnicalData", and "Documentation".
- Center Pane:** Displays the content of the selected package. It shows a list of submodels and their properties. For example, the "World" submodel has properties like "Position\_X", "Position\_Y", "Position\_Z", "Position\_U", "Position\_V", and "Position\_W".
- Right Pane:** Shows the "Content" tab for the selected package. It displays the "Submodel Element (Property)" and its details, including "Referable", "Kind", "Semantic ID", "Qualifiable", "HasDataSpecification", "Data Specification Content", and "Property".

The status bar at the bottom indicates that the package is loaded successfully: "AASX C:\ProjectW0. 스마트공장 파일럿W0527교육W실습용 AASX 파일W완성본W\Nestfield.xml loaded." and "0 bytes No errors Clear Report".

# 작성된 AAS를 솔루션에 적용하는 절차

- 2단계 – Auto-MOS 깃허브에 등록된 AAS 파서 사용하여 설정파일 변환

The image shows the GitHub repository for 'AAS-Parser-for-Windows' and a screenshot of its web-based application interface. The GitHub page on the left lists files like 'AAS\_Parser\_V20230127.zip' and 'README.md'. The application interface on the right is titled 'AAS Parser for AAS-based cloud data acquisition and storage system' and includes sections for 'Introduction', 'Running on Windows', 'Function Button', and 'Questions and Answers'. Green boxes and arrows highlight specific steps: 'Step1-1. Choose a file' points to the '파일 선택' button; 'Step1-2. Click the upload button' points to the 'upload' button; 'Step2. Click the converting button' points to the 'CONVERT' button; 'Step3. View the converted files' points to the 'OUTPUT LOG' section; and 'Step4. Download the converted files' points to the 'DOWNLOAD' section.

**0. AAS 작성 시 참고사항**

OPCUA 태그 정보를 이용하여 AAS 작성 시, 데이터 타입은 아래와 같이 변환되니 참고하여 작성바랍니다.

[AAS Datatype -> OPCUA Datatype]

nonNegativeInteger -> UInt32  
unsignedLong -> UInt64  
unsignedShort -> UInt16  
Byte/unsignedByte -> Byte  
Integer/Int -> Int32  
Short -> Int16

**1. Introduction**

This program is used to convert the AAS files into a suitable format for AAS-based cloud data acquisition & storage system and MOS(Manufacturing Operation System), and was developed to be used in Windows 10 (and 11) environment to provide the user convenience.

**2. Installation**

**Step 0**

- Put the downloaded files in C drive.

**AAS Parser for AAS-based cloud data acquisition and storage system**

**Introduction**

This program is used to convert the AAS files into a suitable format for AAS-based cloud data acquisition and storage system, and was developed to be used in Windows 10 (and 11) environment to provide the user convenience.

[NOTE] You can see a more detailed guide here: <https://github.com/auto-mos>

**Running on Windows**

Firstly, convert the extension(.aasx) of an AAS model to XML using the AASX Package Explorer. Then you upload the converted XML file into the AAS directory(C:\AAS\_Parser\_V20230105\Aas). Upload the XML file using the two buttons below.

파일 선택    선택된 파일 없음    upload

**Function Button**

**CONVERT**

Converting your AAS model

**OUTPUT LOG**

viewing 'engineering.csv'  
viewing 'syscfg.json'

**DOWNLOAD**

engineering.csv  
syscfg.json  
nodeset.xml

**Questions and Answers**

Q. AAS Conversion files are not created.

A. 1. If the IRI of CloudDataSolution AAS and the IRI of EdgeSolution Submodel are different from the ones below, modify them and try conversion again. You can download the AAS template here: <https://github.com/auto-mos>

- CloudDataSolution Submodel IRI: <http://www.aasnest.io/ids/sm/CloudDataSolution>  
- EdgeGWSolution Submodel IRI: <http://www.aasnest.io/ids/sm/EdgeGWSolution>

A. 2. In the AAS model, check the values of IdShort, Value, and ValueType of each element.



# 작성된 AAS를 솔루션에 적용하는 절차

- 3단계 – Github 페이지에 안내된 방법에 따라 내용 수정하여 MOS Cloud 업로드

※ 업로드 경로 : MOS Cloud의 /opt/cfg

• This file is for mapping field data tags with AAS tags.

A	B	C	D	E	F
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_1	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	0
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_2	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	1
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_3	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	2
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_4	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	3
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_5	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	4
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_6	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	5
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_7	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	6
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_8	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	7
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_9	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	8
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_10	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	9
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_11	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	10
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_12	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	11
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_13	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	12
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_14	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	13
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_15	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	14
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_16	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	15
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_17	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	16
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_18	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	17
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_19	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	18
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_20	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	19
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_21	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	20
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_22	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	21
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_23	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	22
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_24	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	23
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_25	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	24
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_26	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	25
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_27	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	26
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_28	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	27
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_29	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	28
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_30	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	29
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_31	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	30
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_32	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	31
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_33	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	32
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_34	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	33
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_35	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	34
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_36	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	35
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_37	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	36
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_38	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	37
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_39	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	38
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_40	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	39
ns=Zs=ProSix6A106F1565.OperationalData.Variable.TestNum_41	MOS_GW	ProductionLine	ns=6s=-Program:TEST_VAR	50	40

- A, E, F Columns are created automatically and B, C, D Columns are needed to be filled manually.
- Details are as follows:
  - A : AAS Tag name from AAS file. This doesn't need to be modified.
  - B : Gateway name. This column should be filled with the gateway name (defined in AAS) from which the data will be collected.
  - C : Device name. This column should be filled with the field device name (defined in AAS) from which the data will be collected.
  - D : Field Tag name. This column should be filled with the field data tag name(OPCUA). It contains namespace and tag identifier.
  - E : Sampling rate. Basic value is 50 and engineer can change this column with appropriate value.
  - F : Array index. If field data tag is an array, engineer should fill this column with array index. But field data tag is not an array, it should be filled with the value : -1(default value).



**Smart  
Manufacturing**

**Digital  
Twin**

**Asset  
Administration  
Shell**

**개정 이력**

문서버전	변경내역
2022-02-09	<ul style="list-style-type: none"> <li>• TYPE-2 인터페이스 : 내용 정리/예제 추가</li> <li>• TYPE-4 인터페이스 : 규정 추가</li> </ul>
2022-02-23	<ul style="list-style-type: none"> <li>• TYPE-2 인터페이스 : 조회시간 지정 없이 저장되어 있는 최종 값을 읽을 수 있도록 API 수정</li> </ul>
2022-03-10	<ul style="list-style-type: none"> <li>• 문서 양식 변경</li> <li>• TYPE-2 인터페이스 : AAS 태그값 변경 API 서비스에 POST 추가</li> </ul>
2022-03-16	<ul style="list-style-type: none"> <li>• AAS 태그명 확인 방법 추가</li> </ul>
2023-11-16	<ul style="list-style-type: none"> <li>• TYPE-2 인터페이스 : 조회 실패 시 응답 및 블록 데이터 값 업데이트 기능 추가</li> <li>• TYPE-4 인터페이스 : Timestamp 일괄/개별 전송 방법 및 예시 추가</li> <li>• AAS의 MOS 적용절차 : Github의 AAS Parser 사용하도록 내용 변경</li> </ul>