NDH Asset Servant 概念澄清

作者: Manus AI
日期: 2025年10月14日
版本: 1.0

核心概念澄清

Asset Servant 的本質

Asset Servant 並非資產本身,而是資產的「數位分身映射層」。

它的核心職責非常明確且簡單:

> ***將 IADL 定義的抽象 Tag 映射到時序資料庫中的實際資料點***

正確的理解

1. Asset Servant 是什麼?

Asset Servant 是一個**輕量級的映射服務實例**,負責:

```
 IADL 中定義的抽象 Tag
         ↓ (映射)
Asset Servant (映射層)
         ↓ (查詢/寫入)
時序資料庫中的實際資料點
```

具體來說:

```yaml
# IADL 定義（抽象層）
asset:
  asset_id: PUMP-001
  data_tags:
    - tag_id: discharge_pressure
      tag_name: "出口壓力"
      data_type: float
      unit: bar
```

Asset Servant 的工作：

```
# Asset Servant 負責映射
{
  "PUMP-001.discharge_pressure" -> "TDengine.plant1.pump001_pressure"
}

# 或映射到 PI System
{
  "PUMP-001.discharge_pressure" -> "PI.PLANT1_PUMP001_DISCH_PRESS"
}
```

2. Asset Servant 不是什麼？

❌ **不是**資產的完整數位分身
❌ **不是**資產的 3D 模型
❌ **不是**資產的業務邏輯執行引擎
❌ **不是**資產的控制系統

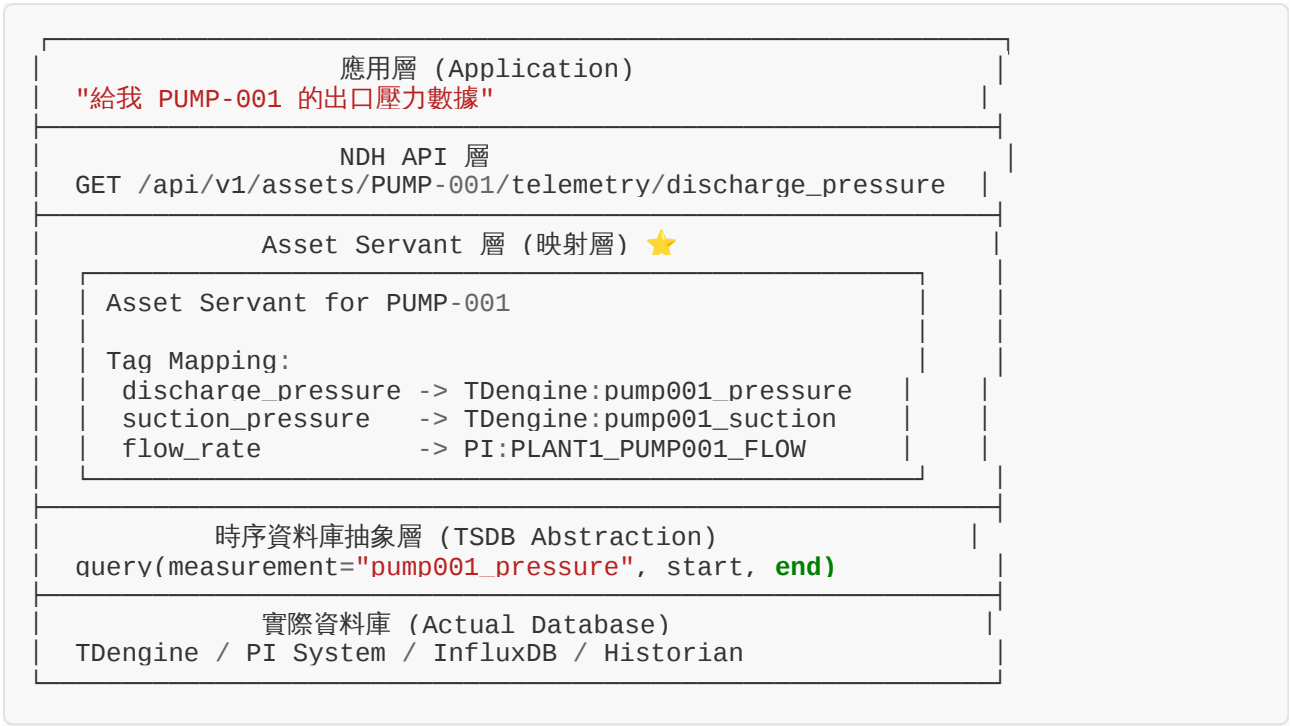
✅ **只是**一個 Tag 映射和數據訪問的代理層

架構關係

完整的數據流

```
物理資產（實際的泵浦）
    ↓（感測器）
PLC / SCADA（採集數據）
    ↓（OPC UA / Modbus）
數據採集層（Connector）
    ↓（寫入）
時序資料庫（TDengine / PI / InfluxDB）
    ↑（映射查詢）
Asset Servant（映射層）⭐
    ↑（API 調用）
NDH API 層
    ↑（HTTP/GraphQL）
應用層（Dashboard / AI）
```

Asset Servant 在架構中的位置



核心功能

Asset Servant 的唯一職責

```python
class AssetServant:
    """資產 Tag 映射服務"""

    def __init__(self, asset_id: str, iadl_definition: dict):
        self.asset_id = asset_id
        self.tag_mappings = self._build_tag_mappings(iadl_definition)

    def _build_tag_mappings(self, iadl: dict) -> dict:
        """
        建立 IADL Tag 到實際資料點的映射

        Returns:
            {
                "discharge_pressure": {
                    "db_type": "tdengine",
                    "measurement": "pump001_pressure",
                    "field": "value"
                },
                "flow_rate": {
                    "db_type": "pi_system",
                    "point_name": "PLANT1_PUMP001_FLOW"
                }
            }
        """
        mappings = {}
        for tag in iadl.get('data_tags', []):
            tag_id = tag['tag_id']
            source = tag.get('source', {})

            if source.get('type') == 'tdengine':
                mappings[tag_id] = {
                    'db_type': 'tdengine',
                    'measurement': source['measurement'],
                    'field': source.get('field', 'value')
                }
            elif source.get('type') == 'pi_point':
                mappings[tag_id] = {
                    'db_type': 'pi_system',
                    'point_name': source['point_name']
                }

        return mappings

    async def get_tag_data(self, tag_id: str, start: datetime, end: datetime):
        """
        獲取指定 Tag 的數據 (透過映射)

        這是 Asset Servant 的核心功能:
        1. 查找 Tag 映射
        2. 路由到正確的資料庫
        3. 返回數據
        """
        if tag_id not in self.tag_mappings:
            raise ValueError(f"Tag {tag_id} not found in asset
{self.asset_id}")

        mapping = self.tag_mappings[tag_id]
```

```
        # 根據映射路由到正確的資料庫
        if mapping['db_type'] == 'tdengine':
            return await self.tdengine_adapter.query(
                measurement=mapping['measurement'],
                start_time=start,
                end_time=end
            )
        elif mapping['db_type'] == 'pi_system':
            return await self.pi_adapter.query(
                measurement=mapping['point_name'],
                start_time=start,
                end_time=end
            )
```

IADL 中的 Tag Source 定義

完整的 IADL 範例

```yaml
# pump001.iadl.yaml

asset:
  asset_id: PUMP-001
  asset_type: centrifugal_pump
  description: "主循環泵浦"

  # 3D 模型 (與 Asset Servant 無關)
  geometry:
    model_file: "models/pump_3d.usd"

  # 數據標籤定義 (Asset Servant 的核心)
  data_tags:
    # Tag 1: 出口壓力 (存在 TDengine)
    - tag_id: discharge_pressure
      tag_name: "出口壓力"
      description: "泵浦出口壓力"
      data_type: float
      unit: bar
      source:
        type: tdengine
        measurement: pump001_pressure
        field: value

    # Tag 2: 入口壓力 (存在 TDengine)
    - tag_id: suction_pressure
      tag_name: "入口壓力"
      data_type: float
      unit: bar
      source:
        type: tdengine
        measurement: pump001_suction
        field: value

    # Tag 3: 流量 (存在 PI System)
    - tag_id: flow_rate
      tag_name: "流量"
      data_type: float
      unit: m3/h
      source:
        type: pi_point
        point_name: PLANT1_PUMP001_FLOW
        pi_server: pi-server.company.com

    # Tag 4: 運行狀態 (存在 Historian)
    - tag_id: running_status
      tag_name: "運行狀態"
      data_type: bool
      source:
        type: historian
        tag_name: PUMP001.STATUS
        historian_server: ge-historian.company.com

  # 屬性 (靜態數據,與 Asset Servant 無關)
  properties:
    manufacturer: "Grundfos"
    model: "CR 64-2"
```

```yaml
    rated_power: 15  # kW
    installation_date: "2020-01-15"
```

Asset Servant 只關心 `data_tags.source`

Asset Servant 在初始化時,會讀取 IADL 的 `data_tags` 部分,建立映射表:

```
{
    "discharge_pressure": {
        "db": "tdengine",
        "measurement": "pump001_pressure"
    },
    "suction_pressure": {
        "db": "tdengine",
        "measurement": "pump001_suction"
    },
    "flow_rate": {
        "db": "pi_system",
        "point": "PLANT1_PUMP001_FLOW"
    },
    "running_status": {
        "db": "historian",
        "tag": "PUMP001.STATUS"
    }
}
```

與其他組件的關係

Asset Servant vs. IADL

組件	職責
IADL	定義資產的「藍圖」(數據模型、3D、屬性)
Asset Servant	執行時的「映射代理」(Tag → 實際資料點)

Asset Servant vs. 時序資料庫

組件	職責
時序資料庫	實際存儲時序數據 (TDengine/PI/InfluxDB)
Asset Servant	提供抽象的 Tag 訪問接口,隱藏底層資料庫細節

Asset Servant vs. 3D 數位分身

組件	職責
3D 數位分身	在 Omniverse 中的視覺化表示
Asset Servant	為 3D 分身提供即時數據 (透過 Tag 映射)

實際應用場景

場景 1: 應用層查詢數據

```python
# 應用層代碼 (不需要知道數據在哪個資料庫)

from ndh.api import NDHClient

client = NDHClient()

# 查詢 PUMP-001 的出口壓力
# Asset Servant 會自動映射到 TDengine 的 pump001_pressure
data = await client.get_asset_telemetry(
    asset_id="PUMP-001",
    tag_id="discharge_pressure",
    start_time="2024-01-01",
    end_time="2024-01-31"
)

# 查詢流量
# Asset Servant 會自動映射到 PI System 的 PLANT1_PUMP001_FLOW
flow_data = await client.get_asset_telemetry(
    asset_id="PUMP-001",
    tag_id="flow_rate",
    start_time="2024-01-01",
    end_time="2024-01-31"
)
```

場景 2: 混合資料源

同一個資產的不同 Tag 可以來自不同的資料庫:

```
PUMP-001:
  - discharge_pressure  → TDengine
  - suction_pressure    → TDengine
  - flow_rate           → PI System
  - running_status      → GE Historian
  - vibration           → InfluxDB
```

Asset Servant 統一管理這些映射,應用層完全不需要關心。

總結

Asset Servant 的正確定位

✅ **是**: 輕量級的 Tag 映射和路由層
✅ **是**: IADL 抽象 Tag 與實際資料點的橋樑
✅ **是**: 多資料源的統一訪問代理

❌ **不是**: 完整的數位分身
❌ **不是**: 業務邏輯執行引擎
❌ **不是**: 資料存儲層

核心價值

1. **抽象化**: 應用層使用語義化的 Tag ID (如 `discharge_pressure`),而非底層的資料點名稱
2. **統一訪問**: 無論數據在 TDengine、PI 還是 Historian,訪問方式完全一致
3. **靈活映射**: 可以輕鬆更改底層資料源,無需修改應用代碼
4. **多源整合**: 同一資產的 Tag 可以分散在不同的時序資料庫中

Asset Servant 就像是一個「智能路由器」,根據 IADL 的定義,將抽象的 Tag 請求路由到正確的實際資料點。