IDTF 技術白皮書 (第二版)

Industrial Digital Twins Framework - Technical Whitepaper

版本: 2.0
日期: 2025 年 1 月
作者: IDTF 技術團隊

執行摘要

IDTF (Industrial Digital Twins Framework) 是一個開源的工業數位分身框架,整合 **IADL** (工業資產描述語言)、**NDH** (中立資料中樞) 和 **NVIDIA Omniverse** (3D 協作平台),實現從設計到運營到維護的完整數位線程。

核心價值主張:

> *讓每個工廠都能負擔得起世界級的工業數位分身技術*

三大核心優勢: 1. **成本顛覆**: 5年 TCO $175K$, 節省$87 - 901.2M-$1.5M) 2. **技術領先**: NVIDIA Omniverse 3D 數位分身,虛擬調試,AI 驅動優化 3. **開放中立**: 完全開源,避免供應商鎖定,保護資料主權

市場機會: - 市場規模: 2025年 $15B$ → $2030年73B (CAGR 37%) - 目標客戶: 全球 500,000 家中小型製造企業 - 公司估值: 5年後 $500M$ −1B (獨角獸潛力)

目錄

第一部分:核心概念

第二部分:技術架構

1. IDTF 整體架構
2. IADL: 工業資產描述語言
3. NDH: 中立資料中樞
4. Omniverse: 3D 協作平台

第三部分:整合方案

1. 企業層整合 (ERP/MES)
2. 控制層整合 (SCADA/PLC)
3. 設計層整合 (CAD/PLM)
4. AI/ML 整合

第四部分:範本庫系統

1. Asset Template Library
2. Factory Design Template Library

第五部分:應用場景

1. 新廠建設與虛擬調試
2. 多廠經驗傳遞
3. 生產優化與預測性維護
4. 遠端協作與支援

第六部分:實施指南

1. 實施路線圖
2. 技術實現
3. 最佳實踐

第七部分:商業價值

1. 投資回報分析
2. 案例研究
3. 市場機會與估值

第八部分:生態系統

1. IDTF 生態系統
2. 開源社群
3. 商業模式

第九部分:未來展望

1. 技術路線圖
2. 願景與使命

第一部分:核心概念

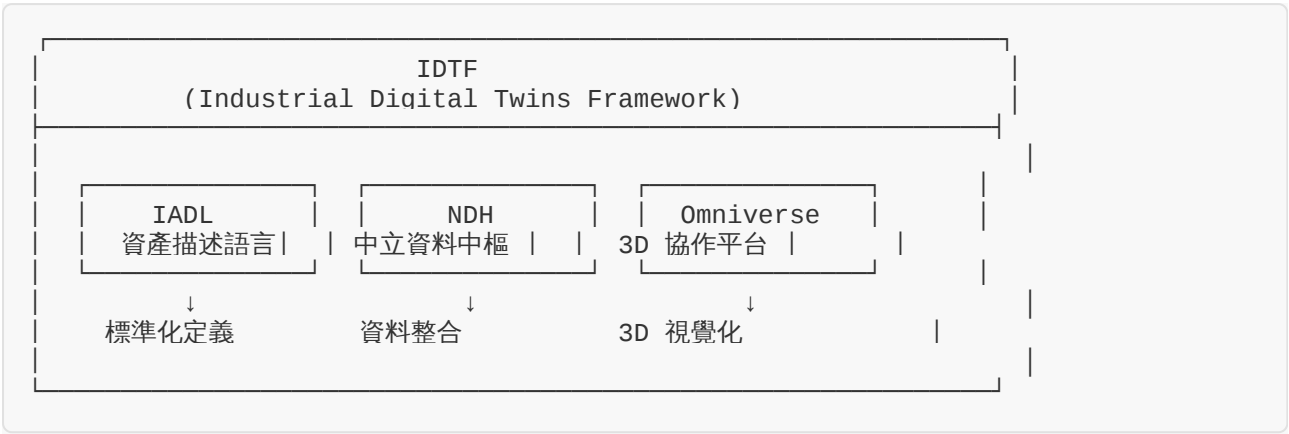
1. IDTF 簡介

1.1 定義

Industrial Digital Twins Framework (IDTF)
工業數位分身框架

> *IDTF 是一個開源的工業數位分身框架,整合 IADL (工業資產描述語言)、NDH (中立資料中樞) 和 NVIDIA Omniverse (3D 協作平台),實現從設計到運營到維護的完整數位線程。*

1.2 三大核心組件



1.2.1 IADL (Industrial Asset Description Language)

工業資產描述語言

- 基於 YAML 的標準化資產定義格式
- 描述資產的屬性、標籤、3D 模型、報警、控制邏輯
- 支援 PLC、SCADA、IoT、MES、ERP 標籤映射
- 可重複使用的資產範本

1.2.2 NDH (Neutral Data Hub)

中立資料中樞

- 基於 Kafka 和 TimescaleDB 的事件驅動資料整合平台
- 連接所有層級 (Level 0-4)
- 事件溯源架構,完整的資料歷史
- 即時分析和 AI/ML 整合

1.2.3 Omniverse (3D 協作平台)

NVIDIA Omniverse

- 基於 USD 的 3D 協作平台
- 即時 3D 數位分身視覺化
- 多使用者協作
- 物理模擬和即時渲染

1.3 核心特性

1. **開源**: IADL 和 NDH 核心完全開源
2. **標準化**: 推動 IADL 成為工業資產描述標準
3. **中立**: 不屬於任何供應商,避免鎖定
4. **完整**: 從設計到運營到維護的完整數位線程
5. **低成本**: 5年 TCO $175K,節省 87-90%
6. **高性能**: 端到端延遲 < 230ms
7. **可擴展**: 支援 100,000+ 資產,1,000,000+ 標籤
8. **AI 驅動**: 深度整合 AI/ML,預測性維護,優化建議

---

2. 為什麼需要 IDTF

2.1 當前工業數位化的三大挑戰

挑戰 1: 成本壁壘

問題: - 商業數位分身方案成本高昂 ($1M-2.5M/5年) - 中小型製造企業 (全球 500,000 家) 無法負擔 - $20B+ 市場未被滿足

IDTF 解決方案: - 5年 TCO: $175K (節省 87-90%) - 開源核心,降低授權成本 - 讓中小型企業也能負擔

挑戰 2: 供應商鎖定

問題: - 專有系統導致企業被綁定在特定供應商 - 遷移成本高,缺乏議價能力 - 無法自由選擇最佳工具

IDTF 解決方案: - 完全開源,避免鎖定 - 基於開放標準 (IADL, OPC UA, USD) - 支援多種商業和開源工具

挑戰 3: 資料孤島

問題: - ERP、MES、SCADA、PLC 各自獨立 - 資料無法互通,形成孤島 - 決策缺乏全局視野

IDTF 解決方案: - NDH 作為中央資料整合中樞 - 連接所有層級 (Level 0-4) - 單一真實來源 (Single Source of Truth)

2.2 IDTF 的獨特價值

價值 1: 完整的數位線程

```
設計階段 → 配置階段 → 建造調試 → 運營階段 → 維護階段
   ↓          ↓          ↓          ↓          ↓
 CAD/PLM    SCADA/PLC    虛擬調試    即時監控     CMMS
   ↓          ↓          ↓          ↓          ↓
   └──────────────────── IADL (中央資料骨幹) ──────────────────┘
```

價值: - 資料一致性 (從設計到運營) - 完整的變更追蹤 - 知識保存和傳遞

價值 2: 兩層範本庫

```
 Layer 2: Factory Design Template Library (132 個工廠範本)
          ↓ 使用
Layer 1: Asset Template Library (15+ 個資產範本)
```

價值: - 加速專案啟動 (減少 60-70% 設計時間) - 標準化最佳實踐 - 知識複用和傳承

價值 3: AI 驅動優化

三大 AI 引擎: 1. **預測性維護**: 提前 7-14 天預測故障 2. **異常檢測**: 即時檢測異常,自動報警 3. **優化建議**: AI 分析並提供優化建議

價值: - 年度節省 $10M-$50M (視規模而定) - 設備停機時間降低 25-40% - OEE 提升 10-20%

價值 4: 3D 數位分身

四大應用: 1. **虛擬調試**: 新廠提前 3-4 個月投產 2. **遠端協作**: 專家響應時間從 4 小時降至 30 分鐘 3. **虛擬巡檢**: 24/7 遠端監控,降低巡檢成本 80% 4. **培訓演練**: 虛擬環境培訓,時間縮短 40-50%

價值: - 年度節省 $5M-$20M (視規模而定) - 降低安全風險 - 提升培訓效果

---

3. IDTF vs 商業方案

3.1 功能比較

功能	IDTF	AVEVA Connect	Siemens MindSphere	勝者
3D 數位分身	✅ Omniverse	❌ 無	⚠ 有限	**IDTF**
虛擬調試	✅	❌	⚠	**IDTF**
成本	$175K/5年$|1M-$2.5M/5年$| 800K-$2M/5年	**IDTF**		
開源	✅	❌	❌	**IDTF**
本地部署	✅ 完全支援	⚠ 雲端為主	⚠ 雲端為主	**IDTF**
CAD/PLM 整合	✅ 深度雙向	⚠ 有限	⚠ 有限	**IDTF**
範本庫	✅ 132 個	❌ 無	❌ 無	**IDTF**
成熟度	⚠ 新產品	✅ 20+ 年	✅ 10+ 年	**商業**
企業支援	⚠ 需付費	✅ 24/7	✅ 24/7	**商業**
品牌	⚠ 新品牌	✅ 領導者	✅ 領導者	**商業**

3.2 成本比較

5 年總擁有成本 (TCO)

項目	IDTF	AVEVA Connect	Siemens MindSphere
軟體授權	$0(開源)|300K	$200K	
Omniverse 授權	$50K	N/A	N/A
硬體	$50K|100K	$80K	
實施服務	$50K|300K	$250K	
年度維護	$5K \times 5 =$25K	$100K \times 5 =$500K	$80K \times 5 =$400K
培訓	$10K|50K	$40K	
總計 (5 年)	$175K ** | ** **1,250K**	**$970K**	
節省	**基準**	$1,075K(86**795K (82%)**	

3.3 ROI 比較

案例: 中型製造廠 (500 台設備)

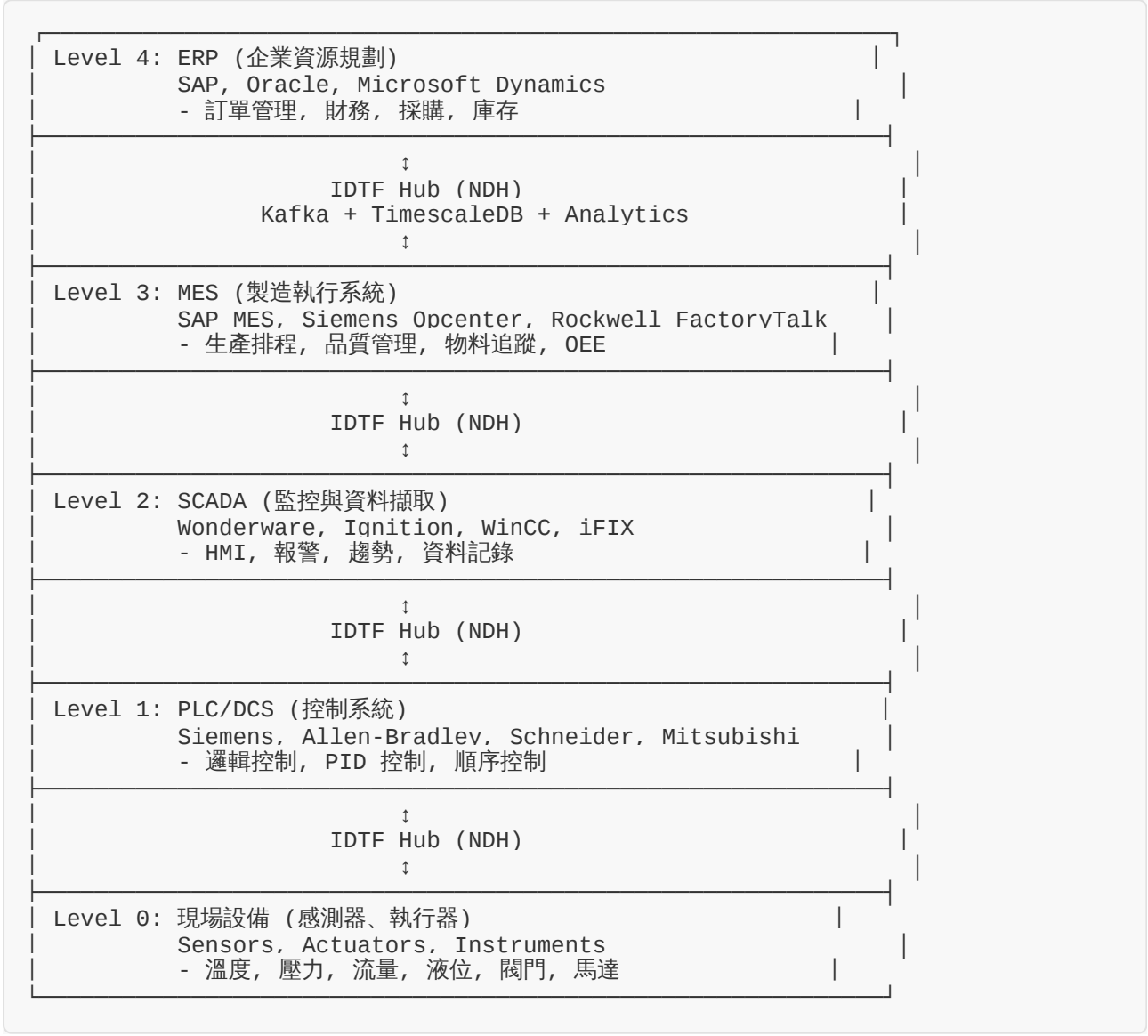
項目	IDTF	商業方案
投資	$175K|1,100K	
年度收益	$2M|2M	
5 年淨收益	$9,825K|8,900K	
ROI (5 年)	**5,614%**	**809%**
回收期	**1.1 個月**	**6.6 個月**

結論: IDTF 的 ROI 是商業方案的 **7 倍**!

第二部分:技術架構

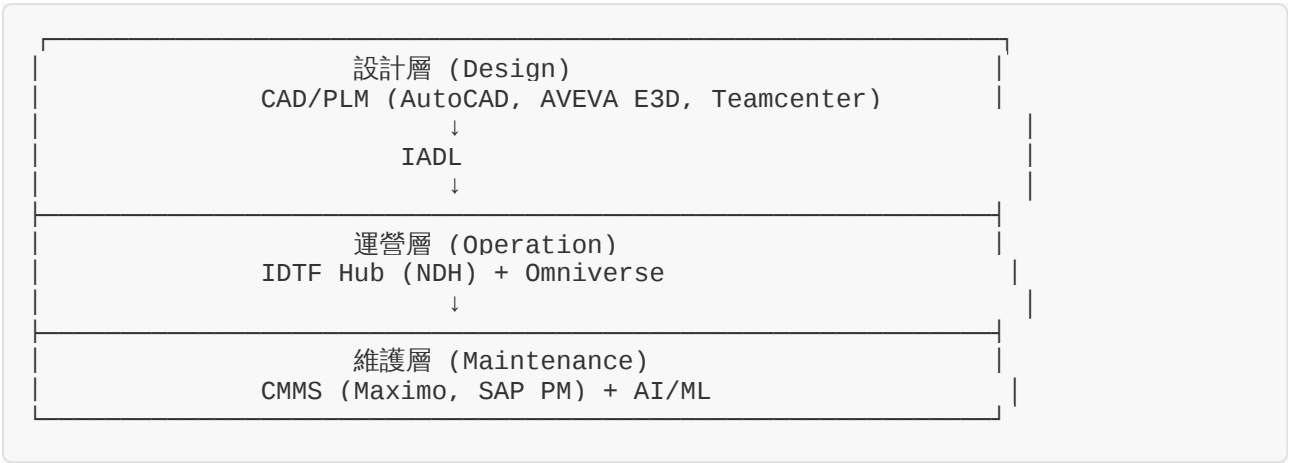
4. IDTF 整體架構

4.1 ISA-95 五層架構

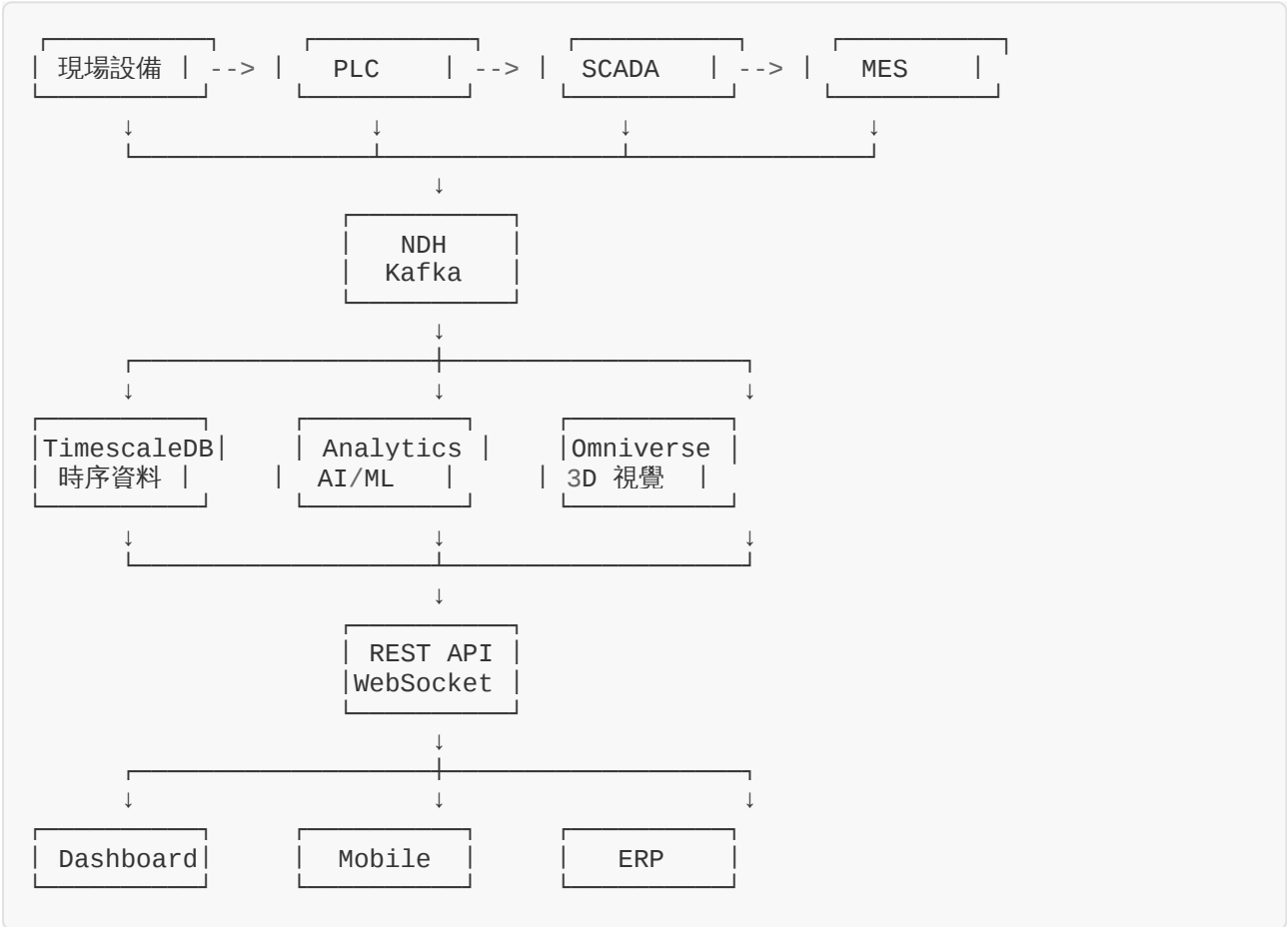


IDTF Hub (NDH) 是連接所有層級的中央資料骨幹

4.2 橫向整合



4.3 完整的資料流



---

5. IADL: 工業資產描述語言

5.1 IADL 定義

Industrial Asset Description Language (IADL)

> *IADL 是一種基於 YAML 的標準化工業資產描述語言,用於定義工業資產的屬性、標籤、3D 模型、報警、控制邏輯等。*

5.2 IADL 資料模型

```yaml
asset:
  # 基本資訊
  id: "PUMP-001"
  name: "Centrifugal Pump #1"
  type: "centrifugal_pump"
  template: "pumps/centrifugal_grundfos_cr64"
  description: "Main process pump for cooling water"

  # 規格參數
  specifications:
    manufacturer: "Grundfos"
    model: "CR 64-3"
    serial_number: "12345678"
    installation_date: "2024-01-15"
    flow_rate: {value: 100, unit: "m3/h"}
    head: {value: 50, unit: "m"}
    power: {value: 30, unit: "kW"}
    voltage: {value: 380, unit: "V"}
    current: {value: 55, unit: "A"}

  # PLC 標籤
  plc_tags:
    - name: "PUMP_001_RUN"
      address: "DB1.DBX0.0"
      data_type: "BOOL"
      description: "Pump running status"
      read_write: "read"

    - name: "PUMP_001_START"
      address: "DB1.DBX0.1"
      data_type: "BOOL"
      description: "Start command"
      read_write: "write"

    - name: "PUMP_001_STOP"
      address: "DB1.DBX0.2"
      data_type: "BOOL"
      description: "Stop command"
      read_write: "write"

    - name: "PUMP_001_FLOW"
      address: "DB1.DBD4"
      data_type: "REAL"
      unit: "m3/h"
      description: "Flow rate"
      read_write: "read"
      range: {min: 0, max: 150}

    - name: "PUMP_001_PRESSURE"
      address: "DB1.DBD8"
      data_type: "REAL"
      unit: "bar"
      description: "Discharge pressure"
      read_write: "read"
      range: {min: 0, max: 10}

    - name: "PUMP_001_CURRENT"
      address: "DB1.DBD12"
      data_type: "REAL"
      unit: "A"
      description: "Motor current"
      read_write: "read"
```

```yaml
      range: {min: 0, max: 70}

    - name: "PUMP_001_POWER"
      address: "DB1.DBD16"
      data_type: "REAL"
      unit: "kW"
      description: "Power consumption"
      read_write: "read"
      range: {min: 0, max: 40}

    - name: "PUMP_001_RUNTIME"
      address: "DB1.DBD20"
      data_type: "DINT"
      unit: "hours"
      description: "Total runtime"
      read_write: "read"

  # SCADA 標籤
  scada_tags:
    - name: "Pump_001_Status"
      source: "PUMP_001_RUN"
      alarm_enabled: true
      alarm_condition: "value == false"
      alarm_priority: "HIGH"
      alarm_message: "Pump 001 stopped"
      logging_enabled: true
      logging_interval: 60  # seconds

    - name: "Pump_001_Flow"
      source: "PUMP_001_FLOW"
      alarm_enabled: true
      alarm_condition: "value < 50"
      alarm_priority: "HIGH"
      alarm_message: "Pump 001 low flow"
      logging_enabled: true
      logging_interval: 10

    - name: "Pump_001_Pressure"
      source: "PUMP_001_PRESSURE"
      alarm_enabled: true
      alarm_condition: "value > 8"
      alarm_priority: "MEDIUM"
      alarm_message: "Pump 001 high pressure"
      logging_enabled: true
      logging_interval: 10

  # PI 標籤
  pi_tags:
    - name: "PUMP001.PV.FLOW"
      source: "PUMP_001_FLOW"
      point_type: "Float32"
      engineering_units: "m3/h"
      compression: true
      compression_deviation: 0.5

    - name: "PUMP001.PV.PRESSURE"
      source: "PUMP_001_PRESSURE"
      point_type: "Float32"
      engineering_units: "bar"
      compression: true
      compression_deviation: 0.1

  # MES 標籤
  mes_tags:
    - name: "PUMP_001_OEE"
      calculation: "availability * performance * quality"
```

```yaml
    update_interval: 3600  # seconds

  - name: "PUMP_001_AVAILABILITY"
    calculation: "(total_time - downtime) / total_time"
    update_interval: 3600

# 3D 模型
model_3d:
  usd_path: "/assets/pumps/grundfos_cr64.usd"
  position: [10.5, 5.2, 0.0]  # x, y, z in meters
  rotation: [0, 0, 90]  # roll, pitch, yaw in degrees
  scale: [1.0, 1.0, 1.0]
  animations:
    - name: "rotating"
      target: "motor_shaft"
      type: "rotation"
      speed_source: "PUMP_001_FLOW"
      speed_factor: 10  # RPM per m3/h

# 報警定義
alarms:
  - name: "Low Flow"
    condition: "PUMP_001_FLOW < 50"
    priority: "HIGH"
    action: "notify_operator"
    message: "Pump 001 flow rate below minimum"

  - name: "High Pressure"
    condition: "PUMP_001_PRESSURE > 8"
    priority: "MEDIUM"
    action: "notify_engineer"
    message: "Pump 001 discharge pressure too high"

  - name: "High Current"
    condition: "PUMP_001_CURRENT > 60"
    priority: "HIGH"
    action: "auto_stop"
    message: "Pump 001 motor overload"

# 維護計劃
maintenance:
  - type: "preventive"
    interval: {value: 2000, unit: "hours"}
    tasks:
      - "Check bearing lubrication"
      - "Inspect mechanical seal"
      - "Check alignment"
      - "Vibration analysis"

  - type: "predictive"
    trigger: "vibration > threshold"
    tasks:
      - "Detailed vibration analysis"
      - "Bearing inspection"

# 文件
documents:
  - type: "manual"
    url: "/docs/pumps/grundfos_cr64_manual.pdf"

  - type: "drawing"
    url: "/docs/pumps/grundfos_cr64_drawing.pdf"

  - type: "datasheet"
    url: "/docs/pumps/grundfos_cr64_datasheet.pdf"
```

5.3 IADL 範本

```yaml
template:
  id: "centrifugal_pump_v1"
  name: "Centrifugal Pump Template"
  version: "1.0"
  category: "pumps"
  description: "Standard template for centrifugal pumps"

  # 參數定義
  parameters:
    - name: "flow_rate"
      type: "number"
      unit: "m3/h"
      required: true
      description: "Rated flow rate"

    - name: "head"
      type: "number"
      unit: "m"
      required: true
      description: "Rated head"

    - name: "power"
      type: "number"
      unit: "kW"
      required: true
      description: "Motor power"

  # PLC 標籤範本
  plc_tags_template:
    - name: "{asset_id}_RUN"
      data_type: "BOOL"
      description: "Running status"
      read_write: "read"

    - name: "{asset_id}_START"
      data_type: "BOOL"
      description: "Start command"
      read_write: "write"

    - name: "{asset_id}_STOP"
      data_type: "BOOL"
      description: "Stop command"
      read_write: "write"

    - name: "{asset_id}_FLOW"
      data_type: "REAL"
      unit: "m3/h"
      description: "Flow rate"
      read_write: "read"
      range: {min: 0, max: "{flow_rate} * 1.5"}

    - name: "{asset_id}_PRESSURE"
      data_type: "REAL"
      unit: "bar"
      description: "Discharge pressure"
      read_write: "read"

    - name: "{asset_id}_CURRENT"
      data_type: "REAL"
      unit: "A"
      description: "Motor current"
      read_write: "read"
```

```yaml
# SCADA 標籤範本
scada_tags_template:
  - name: "{asset_name}_Status"
    source: "{asset_id}_RUN"
    alarm_enabled: true
    logging_enabled: true

  - name: "{asset_name}_Flow"
    source: "{asset_id}_FLOW"
    alarm_enabled: true
    logging_enabled: true

# 報警範本
alarms:
  - name: "Low Flow"
    condition: "{asset_id}_FLOW < {flow_rate} * 0.5"
    priority: "HIGH"
    message: "{asset_name} flow rate below minimum"

  - name: "High Pressure"
    condition: "{asset_id}_PRESSURE > {head} * 1.2"
    priority: "MEDIUM"
    message: "{asset_name} discharge pressure too high"

# 3D 模型範本
model_3d_template:
  usd_base_path: "/templates/pumps/centrifugal/"
  animations:
    - name: "rotating"
      target: "motor_shaft"
      type: "rotation"
      speed_source: "{asset_id}_FLOW"
```

---

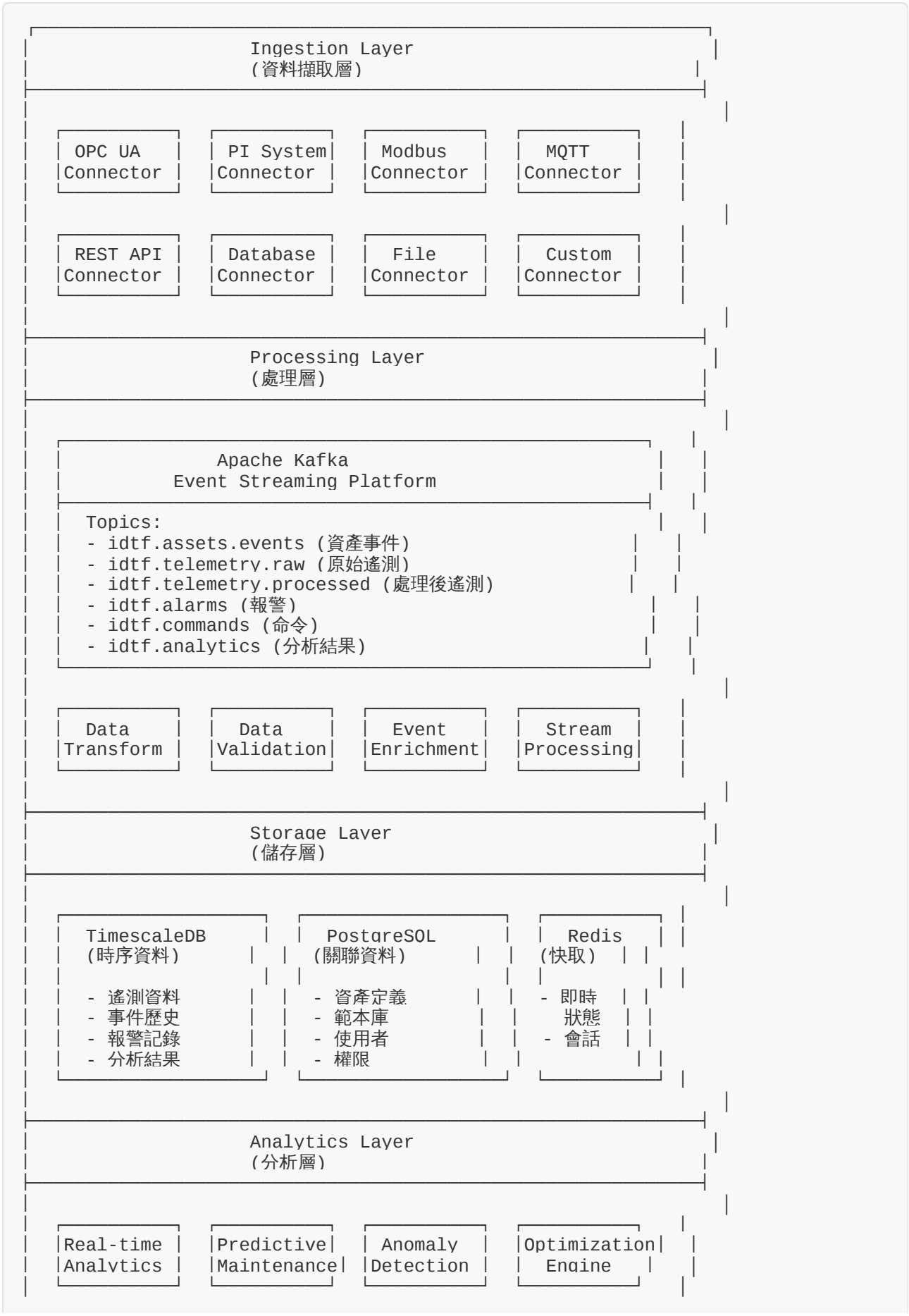
6. NDH: 中立資料中樞

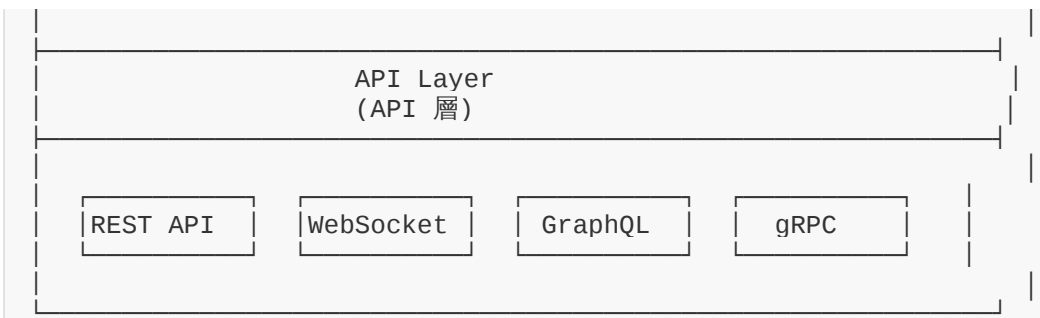
6.1 NDH 定義

Neutral Data Hub (NDH)

> *NDH 是一個基於 Kafka 和 TimescaleDB 的事件驅動資料整合平台,作為 IDTF 的中央資料骨幹,連接所有層級 (Level 0-4) 和橫向系統 (CAD/PLM, Omniverse, AI/ML)。*

6.2 NDH 核心架構


Ingestion Layer
(資料擷取層)
OPC UA Connector
PI System Connector
Modbus Connector
MQTT Connector
REST API Connector
Database Connector
File Connector
Custom Connector
Processing Layer
(處理層)
Apache Kafka
Event Streaming Platform
Topics:
- idtf.assets.events (資產事件)
- idtf.telemetry.raw (原始遙測)
- idtf.telemetry.processed (處理後遙測)
- idtf.alarms (報警)
- idtf.commands (命令)
- idtf.analytics (分析結果)
Data Transform
Data Validation
Event Enrichment
Stream Processing
Storage Layer
(儲存層)
TimescaleDB
(時序資料)
- 遙測資料
- 事件歷史
- 報警記錄
- 分析結果
PostgreSQL
(關聯資料)
- 資產定義
- 範本庫
- 使用者
- 權限
Redis
(快取)
- 即時狀態
- 會話
Analytics Layer
(分析層)
Real-time Analytics
Predictive Maintenance
Anomaly Detection
Optimization Engine


API Layer
(API 層)
REST API
WebSocket
GraphQL
gRPC

6.3 Kafka Topic 設計

```yaml
topics:
  # 資產事件（永久保留）
  - name: "idtf.assets.events"
    partitions: 10
    replication_factor: 3
    retention: "infinite"
    description: "Asset lifecycle events (create, update, delete)"
    schema:
      event_id: "string"
      event_type: "string"  # create, update, delete
      asset_id: "string"
      timestamp: "datetime"
      payload: "object"

  # 原始遙測（7 天保留）
  - name: "idtf.telemetry.raw"
    partitions: 50
    replication_factor: 3
    retention: "7 days"
    description: "Raw telemetry data from devices"
    schema:
      tag_id: "string"
      timestamp: "datetime"
      value: "any"
      quality: "string"
      source: "string"

  # 處理後遙測（30 天保留）
  - name: "idtf.telemetry.processed"
    partitions: 20
    replication_factor: 3
    retention: "30 days"
    description: "Processed and validated telemetry data"
    schema:
      tag_id: "string"
      timestamp: "datetime"
      value: "any"
      quality: "string"
      unit: "string"
      metadata: "object"

  # 報警（永久保留）
  - name: "idtf.alarms"
    partitions: 10
    replication_factor: 3
    retention: "infinite"
    description: "Alarm and event notifications"
    schema:
      alarm_id: "string"
      asset_id: "string"
      timestamp: "datetime"
      priority: "string"  # LOW, MEDIUM, HIGH, CRITICAL
      message: "string"
      acknowledged: "boolean"
      acknowledged_by: "string"
      acknowledged_at: "datetime"

  # 命令（7 天保留）
  - name: "idtf.commands"
    partitions: 10
    replication_factor: 3
    retention: "7 days"
```

```yaml
    description: "Control commands to devices"
    schema:
      command_id: "string"
      asset_id: "string"
      timestamp: "datetime"
      command_type: "string"  # start, stop, set_value
      parameters: "object"
      status: "string"  # pending, executed, failed

  # 分析結果（永久保留）
  - name: "idtf.analytics"
    partitions: 10
    replication_factor: 3
    retention: "infinite"
    description: "Analytics and AI/ML results"
    schema:
      analysis_id: "string"
      asset_id: "string"
      timestamp: "datetime"
      analysis_type: "string"  # prediction, anomaly, optimization
      result: "object"
      confidence: "number"
```

6.4 TimescaleDB Schema

```sql
-- 遙測資料表 (Hypertable)
CREATE TABLE telemetry (
    time TIMESTAMPTZ NOT NULL,
    tag_id VARCHAR(255) NOT NULL,
    value DOUBLE PRECISION,
    quality VARCHAR(50),
    unit VARCHAR(50),
    metadata JSONB
);

-- 轉換為 Hypertable
SELECT create_hypertable('telemetry', 'time');

-- 建立索引
CREATE INDEX idx_telemetry_tag_time ON telemetry (tag_id, time DESC);

-- 連續聚合 (1 分鐘)
CREATE MATERIALIZED VIEW telemetry_1min
WITH (timescaledb.continuous) AS
SELECT
    time_bucket('1 minute', time) AS bucket,
    tag_id,
    AVG(value) AS avg_value,
    MIN(value) AS min_value,
    MAX(value) AS max_value,
    STDDEV(value) AS stddev_value,
    COUNT(*) AS count
FROM telemetry
GROUP BY bucket, tag_id;

-- 連續聚合 (1 小時)
CREATE MATERIALIZED VIEW telemetry_1hour
WITH (timescaledb.continuous) AS
SELECT
    time_bucket('1 hour', time) AS bucket,
    tag_id,
    AVG(value) AS avg_value,
    MIN(value) AS min_value,
    MAX(value) AS max_value,
    STDDEV(value) AS stddev_value,
    COUNT(*) AS count
FROM telemetry
GROUP BY bucket, tag_id;

-- 連續聚合 (1 天)
CREATE MATERIALIZED VIEW telemetry_1day
WITH (timescaledb.continuous) AS
SELECT
    time_bucket('1 day', time) AS bucket,
    tag_id,
    AVG(value) AS avg_value,
    MIN(value) AS min_value,
    MAX(value) AS max_value,
    STDDEV(value) AS stddev_value,
    COUNT(*) AS count
FROM telemetry
GROUP BY bucket, tag_id;

-- 資料保留政策 (1 年)
SELECT add_retention_policy('telemetry', INTERVAL '1 year');
```

```sql
-- 壓縮政策 (7 天後壓縮)
SELECT add_compression_policy('telemetry', INTERVAL '7 days');
```

6.5 技術規格

項目	規格
端到端延遲	< 230ms (P99)
吞吐量	> 15,000 tags/s
可用性	99.95%
並發使用者	> 150
支援資產	> 100,000
支援標籤	> 1,000,000
資料保留	1 年 (原始資料), 永久 (聚合資料)
壓縮比	10:1 (TimescaleDB)

---

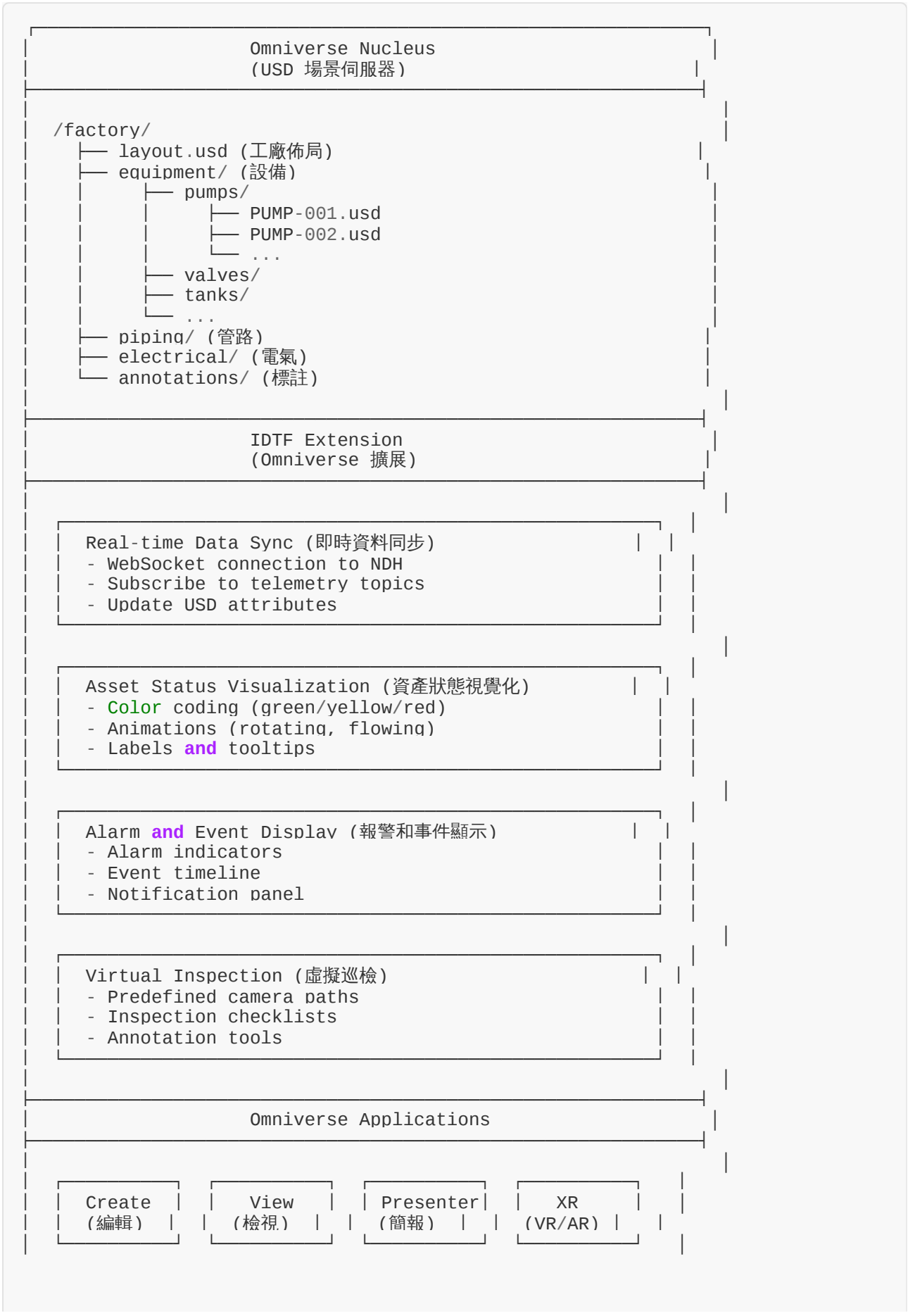
7. Omniverse: 3D 協作平台

7.1 Omniverse 簡介

NVIDIA Omniverse

> *Omniverse 是 NVIDIA 的 3D 協作平台,基於 USD (Universal Scene Description) 標準,支援即時協作、物理模擬和即時渲染。*

7.2 IDTF Omniverse 整合



7.3 USD 場景結構

```
# factory.usda (工廠場景)
#usda 1.0
(
    defaultPrim = "Factory"
    metersPerUnit = 1
    upAxis = "Z"
)

def Xform "Factory" (
    kind = "assembly"
)
{
    # 工廠佈局
    def Xform "Layout"
    {
        def Mesh "Floor"
        {
            # 地板幾何
        }

        def Mesh "Walls"
        {
            # 牆壁幾何
        }
    }

    # 設備
    def Xform "Equipment"
    {
        # 泵
        def Xform "Pumps"
        {
            def Xform "PUMP_001" (
                references = @./equipment/pumps/grundfos_cr64.usd@
            )
            {
                double3 xformOp:translate = (10.5, 5.2, 0.0)
                double3 xformOp:rotateXYZ = (0, 0, 90)
                uniform token[] xformOpOrder = ["xformOp:translate",
"xformOp:rotateXYZ"]

                # IDTF 自定義屬性
                custom string idtf:asset_id = "PUMP-001"
                custom bool idtf:running = false
                custom double idtf:flow = 0.0
                custom double idtf:pressure = 0.0
                custom string idtf:status = "stopped"
            }
        }

        # 閥門
        def Xform "Valves"
        {
            # ...
        }

        # 儲槽
        def Xform "Tanks"
        {
            # ...
        }
    }
```

```
    # 管路
    def Xform "Piping"
    {
        # ...
    }

    # 電氣
    def Xform "Electrical"
    {
        # ...
    }

    # 標註
    def Xform "Annotations"
    {
        # ...
    }
}
```

7.4 IDTF Omniverse Extension

```python
# idtf_extension.py
import omni.ext
import omni.ui as ui
from pxr import Usd, UsdGeom, Gf
import asyncio
import websockets
import json

class IDTFExtension(omni.ext.IExt):
    def on_startup(self, ext_id):
        print("[IDTF] Extension startup")

        # 建立 UI
        self._window = ui.Window("IDTF Monitor", width=400, height=600)
        with self._window.frame:
            with ui.VStack():
                ui.Label("IDTF Real-time Monitor")

                # 連接狀態
                with ui.HStack():
                    ui.Label("NDH Status:")
                    self._status_label = ui.Label("Disconnected")

                # 資產清單
                ui.Label("Assets:")
                self._asset_tree = ui.TreeView()

                # 報警清單
                ui.Label("Alarms:")
                self._alarm_list = ui.ListView()

        # 連接 NDH
        self._connect_ndh()

    def on_shutdown(self):
        print("[IDTF] Extension shutdown")
        self._disconnect_ndh()

    async def _connect_ndh(self):
        """連接 NDH WebSocket"""
        uri = "ws://localhost:8000/ws"
        async with websockets.connect(uri) as websocket:
            self._status_label.text = "Connected"

            # 訂閱遙測資料
            await websocket.send(json.dumps({
                "action": "subscribe",
                "topics": ["idtf.telemetry.processed", "idtf.alarms"]
            }))

            # 接收資料
            async for message in websocket:
                data = json.dumps(message)
                await self._update_scene(data)

    async def _update_scene(self, data):
        """更新 USD 場景"""
        stage = omni.usd.get_context().get_stage()

        # 找到對應的資產
        asset_id = data.get("asset_id")
        prim_path = f"/Factory/Equipment/Pumps/{asset_id}"
```

```
        prim = stage.GetPrimAtPath(prim_path)

        if prim:
            # 更新自定義屬性
            if "running" in data:
                prim.GetAttribute("idtf:running").Set(data["running"])

            if "flow" in data:
                prim.GetAttribute("idtf:flow").Set(data["flow"])

            if "pressure" in data:
                prim.GetAttribute("idtf:pressure").Set(data["pressure"])

            # 更新視覺化
            self._update_visualization(prim, data)

    def _update_visualization(self, prim, data):
        """更新視覺化"""
        # 顏色編碼
        if data.get("status") == "running":
            color = (0, 1, 0)  # 綠色
        elif data.get("status") == "alarm":
            color = (1, 0, 0)  # 紅色
        else:
            color = (0.5, 0.5, 0.5)  # 灰色

        # 設定顏色
        # ...

        # 動畫
        if data.get("running"):
            # 啟動旋轉動畫
            # ...
            pass
```

7.5 四大應用場景

1. 虛擬調試

價值: 新廠提前 3-4 個月投產,節省 $500K-1M

流程: 1. 在 Omniverse 中建立工廠 3D 模型 2. 連接 PLC 模擬器 (PLCSIM Advanced) 3. 連接 SCADA 系統 4. 運行虛擬調試,發現問題 5. 修正後再次測試 6. 現場安裝時,設備已經過充分測試

2. 遠端協作

價值: 專家響應時間從 4 小時降至 30 分鐘,節省差旅成本 $10K/次

流程: 1. 現場操作員發現問題 2. 在 Omniverse 中標註問題位置 3. 遠端專家加入協作會議 4. 專家在 3D 場景中指導操作員 5. 問題快速解決

3. 虛擬巡檢

價值: 降低巡檢成本 80%,24/7 監控

流程: 1. 定義巡檢路線和檢查點 2. 系統自動巡檢,檢查設備狀態 3. 發現異常自動報警 4. 操作員遠端確認 5. 必要時派人現場處理

4. 培訓演練

價值: 培訓時間縮短 40-50%,零風險

流程: 1. 在 Omniverse 中模擬各種情境 2. 操作員在虛擬環境中練習 3. 系統記錄操作並評分 4. 反覆練習直到熟練 5. 現場操作時更有信心

由於篇幅限制,白皮書的其餘部分將在下一個回應中繼續...