# OPC UA がわかる！
# 導入の基本と成功のヒント
# ホワイトペーパー

版数 1.1

作成日: 2025/02/12
更新日: 2026/03/26
更新者: 日本 OPC 協議会

# Contents



# Figures

# Tables

# 1　はじめに

産業のデジタル化が進展する中で、製造現場やインフラ設備から得られるデータの利活用が新たな価値を生み出す基盤となりつつあります。こうした背景において、機器やシステム間での安全かつ柔軟なデータ通信を実現する共通基盤として「OPC UA（Unified Architecture）」が注目されています。

本書は、日本 OPC 協議会 技術部会の活動の一環として、OPC UA の導入を検討する企業・技術者の皆様に向けて、初期段階での理解や設計検討を支援することを目的としています。OPC UA の基本的な考え方や技術的特徴、導入による利点、留意すべきポイントなどを体系的に整理し、現場での実践に役立つ情報を提供します。

## 1.1　OPC UA とは?

OPC UA（OPC Unified Architecture）は、産業用システム間の相互接続と情報交換を実現するために設計されたオープンプラットフォーム型の産業通信規格です。

「OPC」は、もともとは OLE for Process Control の略称として誕生し、Microsoft の COM/DCOM 技術を基盤とした仕様として利用されてきました。しかし、近年の技術的要請やプラットフォームの多様化を背景に、現在では Open Platform Communications という名称へと再定義されています。これにより、特定のベンダーや OS に依存しない、真にオープンな通信仕様として位置づけられています。

OPC UA は、この新たな OPC の方向性を体現するアーキテクチャであり、次のような技術的特徴を備えています。

**表 1　OPC UA の特長**

| 特長 | 説明 |
|---|---|
| オブジェクト指向ベースの構造 | OPC UA は、データを単なる数値として扱うのではなく、構造化された Node（ノード）として表現します。ノードには属性や相互の関係性（Reference）を定義でき、実世界の機器やプロセスを仮想空間上に忠実に表現できます。 |
| アドレス空間と情報モデル | OPC UA は、通信に必要なデータ構造を「意味（セマンティクス）」として定義することが可能です。業界ごとの共通仕様（OPC UA Companion Specification）に基づく情報モデルにより、相互運用性と拡張性の高いデータ連携が実現されます。 |
| セキュリティ設計の重視 | 暗号化、認証、アクセス制御といったセキュリティ機能が標準で実装されており、安全な産業通信を前提とした設計思想に基づいています。 |
| 複数の通信モデルへの対応 | Client/Server モデルに加え、PubSub モデルにも対応しており、リアルタイム性やスケーラビリティの要求にも応えられる柔軟な通信基盤となっています。 |
| 異種環境間での接続性 | 組込み機器、制御システム、IT システム、クラウド環境まで、異なるレイヤー間を同一プロトコルで接続できるため、IIoT やエッジコンピューティングといった先進的な活用にも適応可能です。 |

セマンティクスとは、データが「何を表しているのか」という意味を、機械が正しく解釈できるように表現する考え方です。
単に数値を送るのではなく、「この値は温度なのか」「どの装置のデータなのか」「単位や扱い方は何か」といった情報をあわせて定義することで、異なる機器やシステム間でも同じ意味としてデータを扱えるようになります。

## 1.2 ホワイトペーパーの目的

本書は、日本 OPC 協議会 技術部会が中心となって作成したものであり、OPC UA の導入を検討する企業や技術者に対して、初期段階での理解促進と判断材料の提供を目的としています。

特に、以下のような課題や疑問を持つ読者にとって有用なガイドとなることを意図しています。

- 「OPC UA と従来技術の違いがわからない」
- 「何から着手すればよいかわからない」
- 「PoC を検討しているが、導入の効果や負荷がイメージできない」
- 「セキュリティ設計（認証/認可、証明書運用、SecurityPolicy 選定等）で何を考慮すべきかわからない」
- 「情報モデル設計（型設計、Namespace/NodeId、互換性・バージョニング等）で何を考慮すべきかわからない」

本書では、OPC UA の基本概念から導入時のメリット、典型的な適用事例、さらに導入プロセスと注意点に至るまで、中立的かつ実践的な観点から解説を行っています。また、企業規模や用途に応じた段階的な導入の進め方についても言及しており、初学者から実務担当者まで幅広く活用いただける構成としています。

## 1.3 読者の対象

本書は、OPC UA の導入を検討している方や将来的な活用を見据えて情報収集を行っている幅広い技術者層を対象としています。読者の立場や役割に応じて得られる知見や活用の視点は異なりますが、共通して「現場での具体的な導入や利活用のイメージを持つ」ことを支援する内容としています。

主な読者層は以下の通りです。

■ OT エンジニア（設備・制御系エンジニア）

| | | |
|---|---|---|
| 主な関心領域 | : | 設備データの取得・伝送、既存制御機器との接続方法、導入後の運用保守 |
| 本書の活用視点 | : | 制御システムに OPC UA を適用する際の導入手順や構成例を通じ、導入イメージの具体化に役立ちます。 |

■ IT インフラ担当者・セキュリティ担当者

| | | |
|---|---|---|
| 主な関心領域 | : | ネットワーク構成、クラウド／オンプレミス連携、サイバーセキュリティ対策 |
| 本書の活用視点 | : | OPC UA における通信モデルやセキュリティ機能の解説により、IT 基盤との統合や安全性評価の参考資料として活用できます。 |

■ 組み込み・ソフトウェア開発者

| | | |
|---|---|---|
| 主な関心領域 | : | OPC UA における通信モデル（Client/Server および PubSub）の理解と実装,サーバー／クライアント構成や PubSub 構成の設計指針、オープンソースツールの活用、情報モデルの定義 |
| 本書の活用視点 | : | プロトタイピングや小規模導入の方法を知ることで、実装に向けた初期検討やツール選定の参考になります。 |

本書ではこれらの異なる視点を意識し、各章で適宜技術的背景の解説と実装のヒントを組み合わせて構成しています。初学者だけでなく、実務に携わるエンジニアやマネジメント層にとっても有益なガイドとなることを目指しています。

# 2　OPC UA の基本概念

本章では、OPC UA の基本的な概念について説明します。OPC UA がどのような特徴を持ち、なぜ必要とされているのか、また既存技術との違いについて解説します。これにより、読者が OPC UA の全体像を把握し、導入の意義を理解できるようにすることを目的としています。

## 2.1　OPC UA の特徴

OPC UA は、産業システムにおけるデータ通信の標準規格として、以下のような特徴を備えています。これらの特徴により、システム間の相互運用性、拡張性、そしてセキュリティが向上します。

### 2.1.1 オブジェクト指向ベースの構造

OPC UA は、オブジェクト指向の考え方に基づいて設計されています。ここでは、オブジェクトモデルとノードの構成について説明します。データと機能をオブジェクトとしてまとめることで、システムを直感的に理解しやすく、効率的に管理できます。

各オブジェクトは Node（ノード）と呼ばれ、Variable（変数）、Method（メソッド）、Event（イベント）などの要素を含みます。これにより、実世界の機器やプロセスを仮想空間に忠実に表現できます。

- オブジェクトモデルの構成

OPC UA はオブジェクト指向の設計を採用しており、これによりデータや機能を効率的に管理できます。オブジェクト指向とは、データとその操作を一つの単位（オブジェクト）として扱う考え方です。例えば、工場の機械をオブジェクトとして考えると、その機械の状態（データ）や操作（機能）を一つのまとまりとして管理できます。

OPC UA では、各オブジェクトは Node（ノード）と呼ばれる単位で構成されます。ノードは、データを持つ Variable（変数）や、操作を行う Method（メソッド）などの要素を含んでいます。これにより複雑なシステムでも直感的に理解しやすくなります。

- Variable（変数）　オブジェクトの状態やプロパティを表します。オブジェクトに関連付けられ、その値はオブジェクトの特性を示します。
- Method（メソッド）　オブジェクトに対して実行できる操作を表します。オブジェクトに関連付けられ、クライアントが呼び出すことができます。
- Event（イベント）　システム内で発生する特定の出来事を表します。オブジェクトはイベントを生成することができ、クライアントはこれらのイベントを監視することができます。

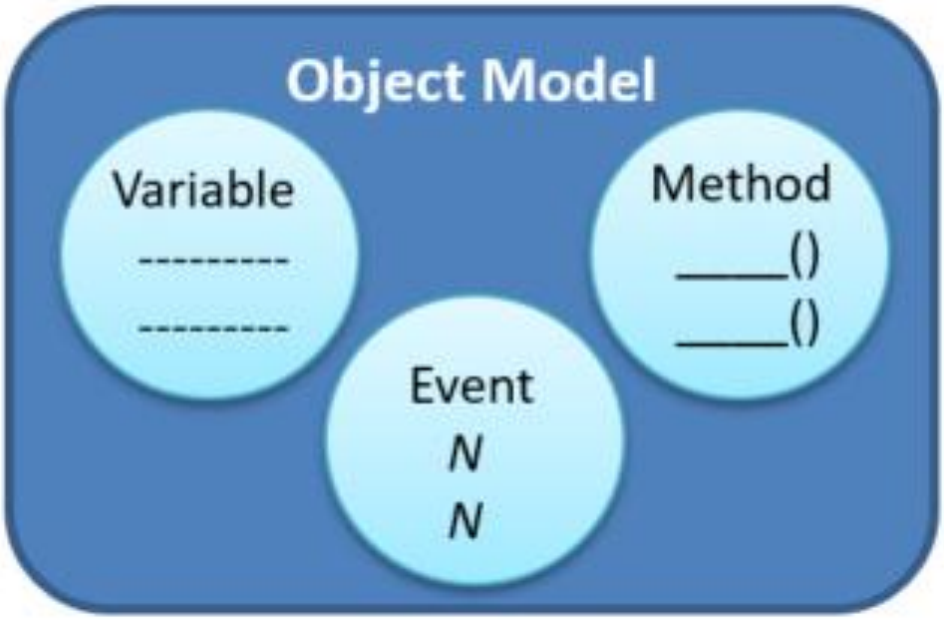



**図 1　オブジェクトモデルの構成**

- Node（ノード）とは

ノードは、OPC UA の世界で、データや情報を構造化して表現する基本的な構成要素です。ノードは NodeId によって一意に管理し、階層的に組織化され、親子関係や参照関係を持つことができます。

各ノードは特定のノードクラスに属し、その役割や性質が定義されています。表 2 に OPC UA で利用できるノードの種類（ノードクラス）を一覧として示します。

**表 2　ノードクラスの一覧**

| ノードクラス | 説明 | 主な用途例 |
|---|---|---|
| Object | 物理的なモノやシステムの一部をオブジェクトとして表現します。Variable（変数）や Method（メソッド）など他のノードを含むことができます。 | 工場のロボット、温度センサ、製造ラインの一部 |
| Variable | データ値を保持するノードです。データ型を持ち、データの読み書きが可能です。リアルタイムで値が変わることがあります。 | 温度、圧力、デバイスの状態などの測定値や設定値 |
| Method | オブジェクトに対して実行できる操作や関数を表します。入力と出力のパラメータを持つことができます。 | デバイスのリセット、データの取得、診断の実行 |
| ObjectType | オブジェクトのテンプレートを定義します。 | 特定の種類の機械の標準的な構造の定義 |
| VariableType | 変数のテンプレートを定義します。 | 特定の種類のセンサの標準的なデータ型や属性の定義 |
| ReferenceType | ノード間の関係の種類を定義します。 | 親子関係や接続関係 |
| DataType | Variable（変数）や Method の入力／出力パラメータで使用されるデータ型を定義します。 | 整数型や浮動小数点型 |
| View | 特定のノードのサブセットを表示するためのフィルターを提供します。 | 特定の用途やユーザー向けにフィルタリングされたノードの表示 |

ノード間の関係は Reference（参照）によって表され、これによりノード同士がリンクされます。さらに、ノードは Attribute（属性）を持ち、名前、データ型、値、アクセス権などの特性やデータを表現します。これにより、実世界の機器やプロセスを仮想空間上に忠実に表現します。

ノードは非常に柔軟で拡張性が高く、ユーザーは自社の業務や装置特性に応じた独自の情報モデルや情報構造を定義することができます。一方で、OPC UA ではコンパニオン仕様や標準情報モデルといった共通の情報モデルが整備されており、これらを基盤として独自モデルを拡張することが可能です。これにより、**共通モデルによる相互運用性を維持しつつ、個別要件への柔軟な対応が可能となります。**

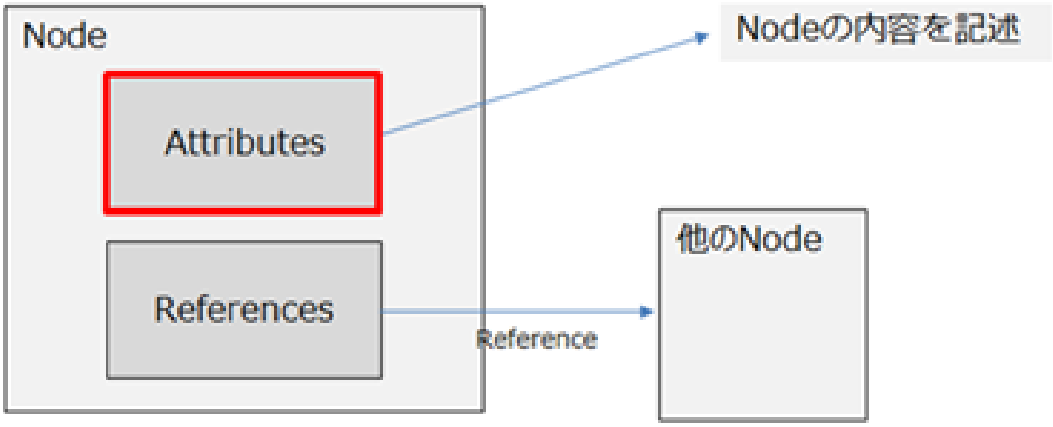



**図 2　ノードとノード間の関係**

- Attribute（属性）とは

ノードに関連付けられたメタデータやプロパティです。一つのノードは複数の属性を持つことができます。ノードの名前、アクセス制御、データ型、値などの情報を提供します。ノードの役割や性質を定義するノードクラスも、属性によって定義します。

表 3 はノードに付加できる主な属性です。

**表 3　主な Attribute（属性）の一覧**

| 属性名 | 説明 |
|---|---|
| NodeId | ノードの一意の識別子。 |
| NodeClass | ノードのクラス（例：Object、Variable、Method など）。 |
| BrowseName | ノードのブラウズ名。 |
| DisplayName | ノードの表示名。 |
| Description | ノードの説明。 |
| WriteMask | ノードの属性がクライアントによって書き込み可能かどうかを示すビットマスク。 |
| UserWriteMask | ユーザーごとの書き込み可能な属性を示すビットマスク。 |
| Value | 変数ノードの現在の値。 |
| DataType | 変数ノードのデータ型。 |
| ValueRank | 変数ノードの値のランク（スカラー、配列など）。 |
| ArrayDimensions | 変数ノードの配列の次元。 |
| AccessLevel | 変数ノードのアクセスレベル（読み取り、書き込みなど） |
| UserAccessLevel | ユーザーごとのアクセスレベル。 |
| MinimumSamplingInterval | 変数ノードの最小サンプリング間隔。 |
| Historizing | 変数ノードが履歴データを保持するかどうか。 |

- References（参照）とは

ノード間の関係を表すリンクです。参照は、あるノードが他のノードとどのように関連付けられているかを示します。これにより、ノード間の構造や階層を定義し、情報モデル全体のナビゲーションを可能にします。

表 4 は、よく利用されるリンクの一覧ですが、ユーザー独自に意味を定義したリンクを作成することも可能です。

**表 4　主な References（参照）の一覧**

| 参照タイプ | 説明 |
|---|---|
| HasComponent | ソースノードがターゲットノードをコンポーネントとして持つことを示します。 |
| HasProperty | ソースノードがターゲットノードをプロパティとして持つことを示します。 |
| Organizes | ソースノードがターゲットノードを組織化することを示します。 |
| HasTypeDefinition | ソースノードがターゲットノードの型定義を持つことを示します。 |
| HasSubtype | ソースノードがターゲットノードのサブタイプであることを示します。 |
| HasModellingRule | ソースノードがターゲットノードのモデリングルールを持つことを示します。 |

### 2.1.2 アドレス空間と情報モデル

ここでは、OPC UA のアドレス空間と情報モデルについて説明します。アドレス空間とは何か、情報モデルとは何か、そして その具体的を示します。

- アドレス空間とは

アドレス空間は、OPC UA サーバー内の構造情報であり、OPC UA サーバーに実際に存在するノードの集合体です。アドレス空間内には、前述のノードが配置されており、これによりデータや機能が一元的に管理されます。例えば、工場全体の機械やセンサの情報を一つのアドレス空間内で管理することができます。

ノードを追加・変更することで、アドレス空間をカスタマイズし、特定のアプリケーションや業界のニーズに対応できます。

- 情報モデルとは

情報モデルは、システム内のデータやリソースをノードという単位で整理し、ノードをどのように設計するかのルールや枠組みです。情報モデルにより、データや機能の構造が明確になり、システム全体の理解が容易になります。例えば、工場の情報モデルでは、機械、センサ、操作などがどのように関連しているかを定義します。

- 情報モデルの具体例

例えば、工場の情報モデルでは、工場オブジェクトノードの下に複数の機械オブジェクトノードがあり、各機械オブジェクトノードの下にセンサオブジェクトノードや変数ノードが配置されます。

図 3 は、工場の情報モデルの例を示しています。

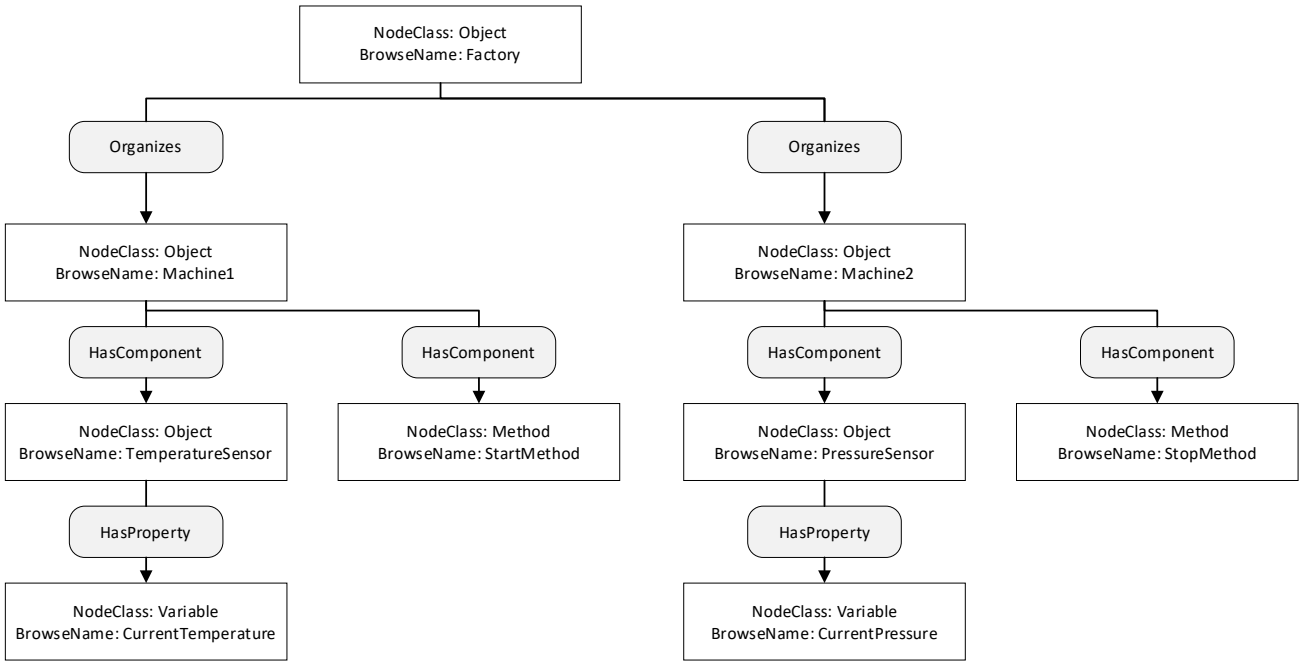



**図 3　工場の情報モデルの例**

表 5 は、 図 3 の各ノードを説明しています。

**表 5　工場の情報モデルの例における各ノード**

| ノード名 | ノードタイプ | 説明 |
|---|---|---|
| Factory（工場） | オブジェクト | 工場全体を表す |
| Machine1（機械 1） | オブジェクト | 工場内の一つの機械を表す |
| Temperature Sensor（温度センサ） | オブジェクト | 機械 1 に取り付けられたセンサを表す |

| ノード名 | ノードタイプ | 説明 |
|---|---|---|
| Current Temperature（現在の温度） | 変数 | 温度センサの現在の温度データ |
| Start Method（開始メソッド） | メソッド | 機械 1 を開始させる操作 |
| Machine2（機械 2） | オブジェクト | 工場内の別の機械を表す |
| Pressure Sensor（圧力センサ） | オブジェクト | 機械 2 に取り付けられたセンサを表す |
| Current Pressure（現在の圧力） | 変数 | 圧力センサの現在の圧力データ |
| Stop Method（停止メソッド） | メソッド | 機械 2 を停止させる操作 |

図 4 は、OPC UA クライアントを用いて OPC UA サーバーのアドレス空間をブラウズした例を示しています。本図は、図 3 で示した「工場の情報モデル例」が、実際の OPC UA サーバー上でどのような階層構造として構成され、クライアントから参照されるかを示したものです。

Root 配下の Objects フォルダ以下に Factory（工場） オブジェクトが配置され、その下に Machine1、Machine2 といった機械オブジェクトが階層的に構成されています。これは、図 3 において Organizes 参照によって Factory が複数の Machine を包含している構造と対応しています。

各 Machine の配下には、HasComponent 参照を用いて Temperature Sensor（温度センサ） や StartMethod（開始メソッド） が配置されています。さらに、Temperature Sensor の下には HasProperty 参照を用いて Current Temperature（現在の温度） が Variable ノードとして定義されており、センサが持つ現在値を表しています。

このように、OPC UA では、図 3 で示した概念的な情報モデル（Factory、Machine、Sensor、Variable、Method）が、OPC UA クライアントからはアドレス空間のツリー構造として可視化されます。これにより、BrowseName や NodeClass（Object、Variable、Method など）といった属性（Attribute）の意味を、実際のアドレス空間の構造と対応付けて理解することができます。

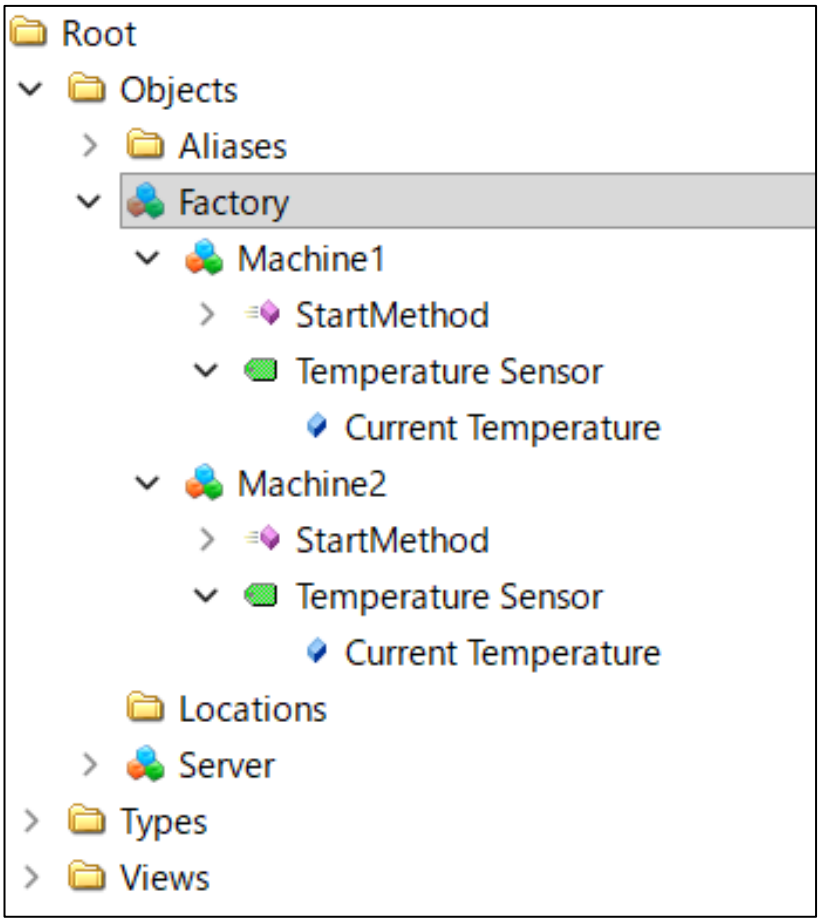



**図 4　OPC UA クライアントでブラウズしたアドレス空間の例（図 3 の工場情報モデルに対応）**

- 情報モデルの拡張性

OPC UA の情報モデルは、ユーザーが特定のニーズに合わせてカスタマイズすることができ（Vendor Specific Extensions）、さらに特定の業界やアプリケーション向けに標準化されたモデルを利用することができます（Companion Information Models）。これにより、システムの柔軟性と相互運用性が大幅に向上します。

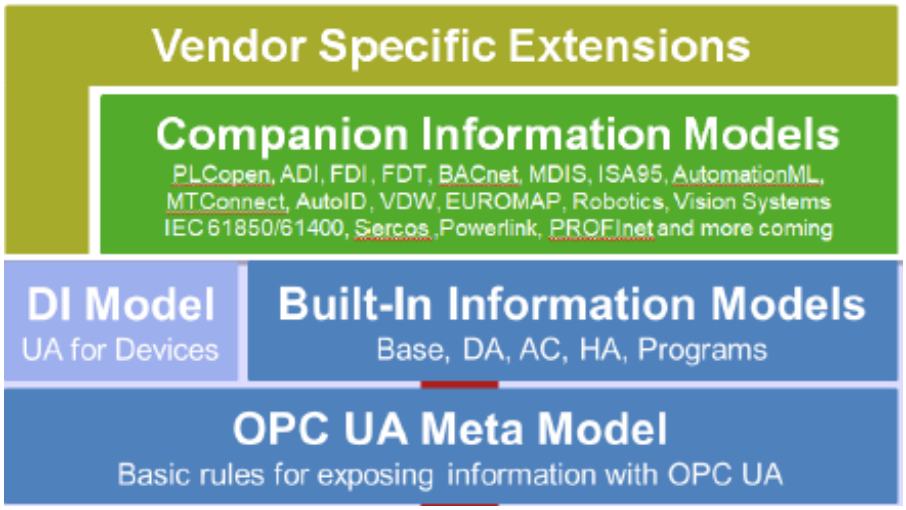



**図 5　情報モデルの拡張性**

図 5 の各要素に関して表 6 で説明します。

**表 6　OPC UA 情報モデルの構成要素と役割**

| 区分 | 図中の表記 | 概要説明 |
|---|---|---|
| OPC UA メタモデル | OPC UA Meta Model | OPC UA における情報表現の最下層となるモデル。ノード、参照、属性、型定義など、情報を OPC UA で公開するための**基本ルールと構造**を定義する。すべての情報モデルはこのメタモデルに基づいて構築される。 |
| 組み込み情報モデル | Built-In Information Models | OPC UA サーバーが共通に備える基本的な情報モデル群。Base、DA（Data Access）、AC（Alarms & Conditions）、HA（Historical Access）、Programs などが含まれ、**データ取得、アラーム通知、履歴参照といった汎用機能**を標準化された形で提供する。 |
| デバイス統合モデル | DI Model（UA for Devices） | 各種機器（デバイス）を OPC UA 上で共通的に扱うための標準情報モデル。デバイス識別情報、状態、パラメータ、構成情報などを定義し、**異なるメーカーや機種であっても共通の方法でデバイスを扱えるようにする**ことを目的とする。 |
| コンパニオン情報モデル | Companion Information Models | 特定の業界や用途向けに標準化された情報モデル。PLCopen、FDI、AutomationML、Robotics、Vision Systems などが該当し、**業界固有の意味（セマンティクス）を共通化**することで、マルチベンダ環境での相互運用性を高める。 |

| | | |
|---|---|---|
| ベンダー固有拡張 | Vendor Specific Extensions | 標準モデルでは表現しきれない独自機能や付加価値を表現するための拡張。ユーザーやベンダーが自由に定義でき、**製品差別化や独自要件への対応**を可能にする。一方で、利用範囲には相互運用性への配慮が必要となる。 |

- アドレス空間と情報モデルの違い

ここでは「アドレス空間」と「情報モデル」の違いを整理した比較表を表 7 に示します。

**表 7　アドレス空間と情報モデルの比較表**

| 項目 | アドレス空間 | 情報モデル |
|---|---|---|
| 定義 | OPC UA サーバー内に実際に存在するノードの集合 | ノードやその関係をどのように設計・定義するかを示す枠組み |
| 役割 | データや機能を保持・管理する「実体」 | ノードの種類・属性・関係性を規定する「設計図」 |
| 例え | 工場そのもの（中に物が配置されている状態） | 工場の棚割り図・配置図（どこに何を置くかのルール） |
| カスタマイズ | ノードを追加・削除・変更することで、実装側で調整可能 | 標準仕様（OPC UA Companion Specification）に従うか、独自に定義して利用 |
| 具体例 | 「工場オブジェクト」配下に「機械ノード」「センサノード」が存在 | 「工場 → 機械 → センサ → 測定値」という階層関係を設計として定義 |
| 発生工程 | OPC UA サーバーの実装時 | OPC UA サーバーの設計時 |

### 2.1.3 セキュリティ設計の重視

OPC UA は、産業オートメーションや IoT システムにおいて高いセキュリティを提供するために、包括的なセキュリティ設計を重視しています。表 8 に、OPC UA のセキュリティ機能について説明します。

**表 8　OPC UA のセキュリティ機能**

| セキュリティ機能 | 説明 |
|---|---|
| 認証 | OPC UA は、アプリケーション認証とユーザー認証の二つのレベルで認証を行います。アプリケーション認証は、通信するアプリケーション間での信頼関係を確立し、ユーザー認証はシステムにアクセスするユーザーの身元を確認します。 |
| 権限 | ユーザー認可は、認証されたユーザーに対して適切なアクセス権限を付与するプロセスです。これにより、ユーザーごとに異なるアクセスレベルを設定し、重要なデータや機能への不正アクセスを防ぎます。 |
| 監視 | 操作の監視は、システム内で行われるすべての操作を記録し、異常な活動や不正アクセスを検出するための重要な機能です。これにより、セキュリティインシデントの早期発見と対応が可能になります。 |
| 暗号化 | OPC UA は、データの機密性を保護するために、公開鍵暗号方式と共通鍵暗号方式を使用します。公開鍵暗号方式は、公開鍵と秘密鍵という対になる 2 つの鍵を使い、共通鍵暗号方式は、暗号化と復号化に同じ 1 つの鍵を使って、データの暗号／復号を行います。<br>また、OPC UA はさまざまな暗号スイートをサポートしており、異なるセキュリティレベルやパフォーマンス要件に応じて適切な暗号スイートを選択できます。これにより、柔軟なセキュリティ設定が可能です。 |
| 署名 | デジタル署名は、データの送信者を確認し、データが改ざんされていないことを保証するために使用されます。これにより、データの信頼性と整合性が確保されます。 |
| メッセージ妥当性の検証 | シーケンス番号の挿入により、メッセージの順序を確認し、再送や重複メッセージを検出します。これにより、通信の信頼性が向上します。 |
| 開示情報の最小化 | ユーザー認可により、必要最低限の情報のみを開示することで、機密情報の漏洩リスクを最小化します。 |
| メッセージの最小化 | 通信データの制限は、必要なデータのみを送信することで、通信量を削減し、セキュリティリスクを低減します。 |
| セッション毎の認証情報 | セッションごとに一意の通信 ID を付与することで、各セッションの認証情報を管理し、不正なセッションの識別と排除が可能になります。 |

### 2.1.4 複数の通信モデルへの対応

ここでは、OPC UA が複数の通信モデルに対応していることについて説明します。具体的には、Client/Server モデルと PubSub モデルの仕組みと特徴について詳しく見ていきます。

- Client/Server モデルの仕組みと特徴

Client/Server モデルは、クライアントがサーバーに対してデータの要求を行い、サーバーがその要求に応じて処理を実行します。例えば、工場の管理システムが機械の状態を確認する際、管理システム（クライアント）が機械の制御システム（サーバー）にデータを要求し、制御システムがそのデータを提供します。

具体的には、クライアントはサーバー内のノードに対して、サーバーが提供する様々なサービスを利用してアクセスします。

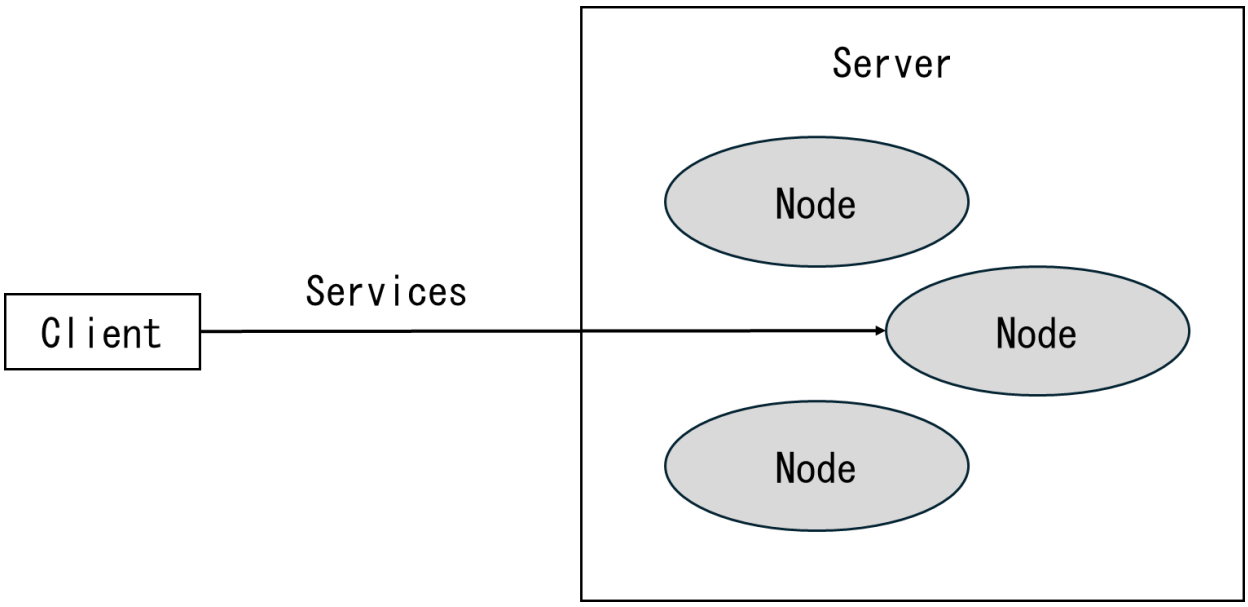



図 6　Client/Server モデル

- PubSub モデルの仕組みと特徴

PubSub モデルは、Publisher がデータを発信し、Subscriber がそのデータを受信します。例えば、工場のセンサが温度データを発信し、管理システムがそのデータを受信する、といった形です。

このモデルの特徴は、通信が非同期であることです。Publisher がデータを発信し続け、Subscriber が必要な時にそのデータを受信します。これにより、リアルタイムでのデータ配信が可能となり、大規模なシステムでも効率的にデータを管理できます。

具体的には、Publisher は以下の流れでデータを配信し、Subscriber はデータを受信します。

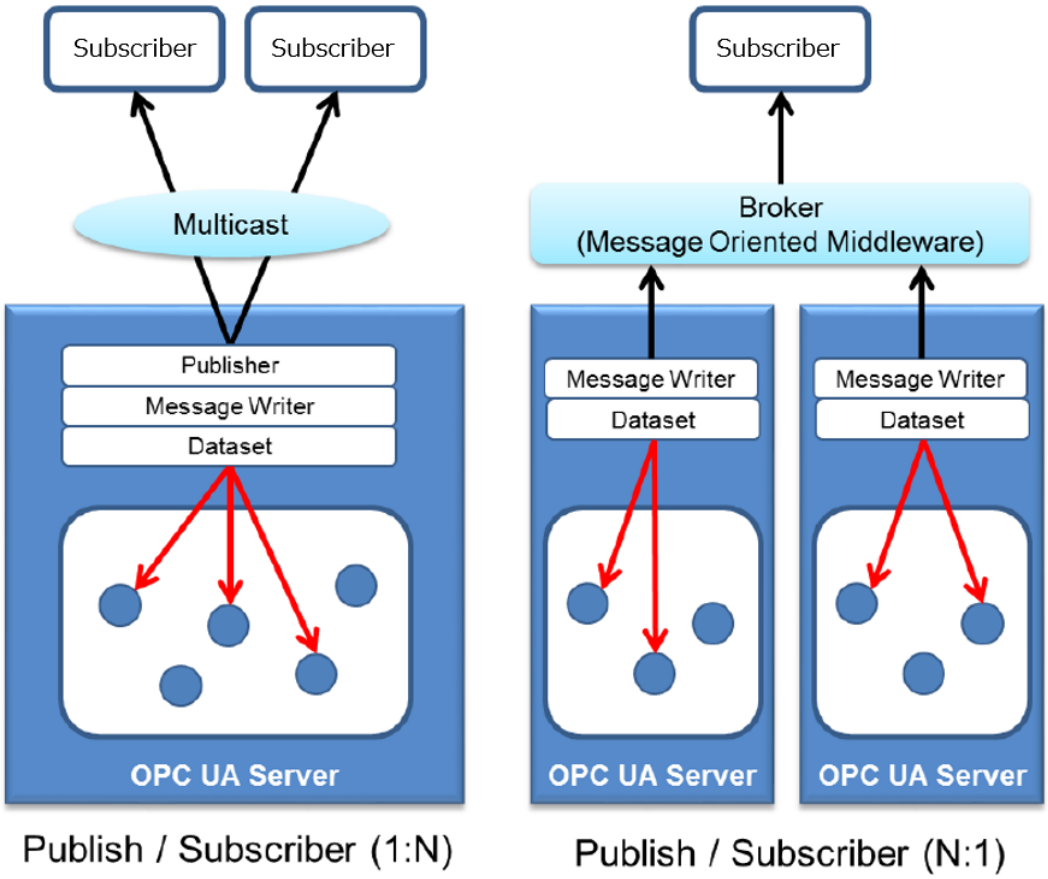



**図 7　PubSub モデル**

＜Publisher 側の動き＞

① データを収集する
② DataSet を作成する
③ MessageWriter が DataSet をエンコードしてメッセージを生成する
④ Publisher がメッセージを配信する（Multicast または Broker 経由）

＜Subscriber の動き＞

※ 図では Subscriber を 1 つの要素として表現しているが、実際には以下の処理を内部で行う。

① Subscriber がメッセージを受信する（Multicast または Broker 経由）
② MessageReader がメッセージをデコードする
③ DataSet（DataSetReader が解釈したデータ）として取り出す
④ アプリケーションがデータを利用する（監視・記録・制御等）

### 2.1.5 異種環境間での接続性

OPC UA は、ISO/OSI 参照モデルにおける第 7 層（アプリケーション層）から第 5 層（セッション層）を中心に仕様化された産業向け通信アーキテクチャです。下位層については、インターネットやローカルネットワークで広く利用されている TCP/IP をはじめ、HTTPS、WebSocket など、既存の標準化された通信プロトコルやネットワーク技術を組み合わせて利用することができます。

データエンコーディングの観点でも、OPC UA はバイナリ、XML、JSON といった複数の形式をサポートしており、異なるシステムやプラットフォーム間でのデータ交換を容易にしています。さらに、OPC UA の開発者コミュニティからは、クロスプラットフォームで利用可能な SDK やライブラリが提供されているため、開発者は特定の OS やハードウェアに依存せずに OPC UA 対応アプリケーションを開発することが可能です。

図 8 は、ISO/OSI 参照モデルと OPC UA の通信方式（Client/Server および PubSub）との概念的な対応関係を示したものです。本図は OPC Foundation が示す代表的な構成例をもとにした概念図であり、各レイヤーへの割り当てを厳密に定義することを目的としたものではありません。OPC UA がどの層の通信手段と組み合わさって動作し得るかを俯瞰的に理解するための参考図として位置付けています。

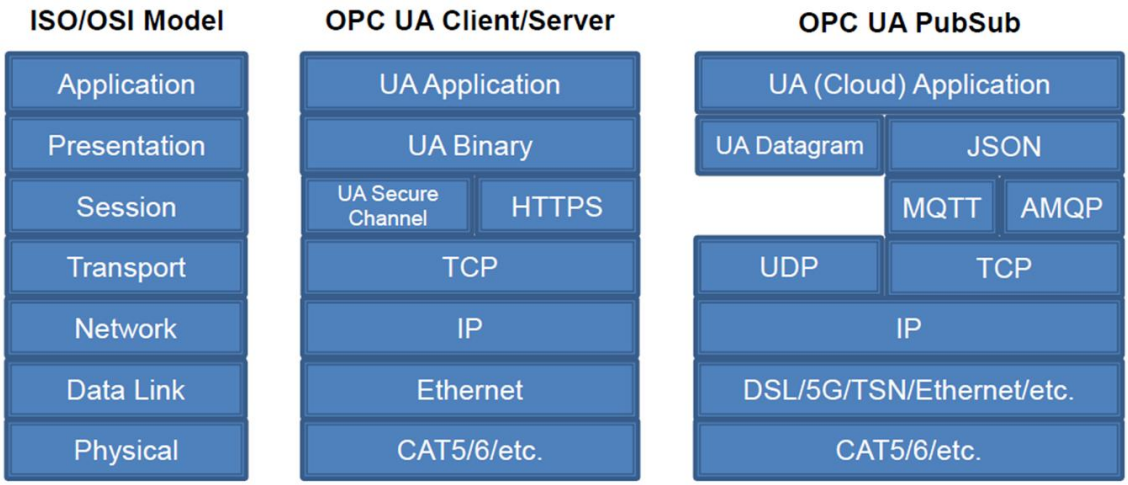



**図 8　ISO/OSI 参照モデルと OPC UA 通信方式（Client/Server・PubSub）の対応関係**

Client/Server と PubSub は上位の通信モデルとして区別されていますが、物理層およびデータリンク層において利用できるネットワーク技術は共通です。例えば、Ethernet に加えて、5G や TSN（Time-Sensitive Networking）といった技術も、通信モデルの別に関わらず利用することが可能です。図 8 において Client/Server と PubSub で異なる表現がなされている場合であっても、それは排他的な対応関係を示すものではありません。

また、5G や TSN は単一の OSI 層に完全に対応付けられるものではなく、複数の層にまたがる技術群として整理されることが一般的です。例えば、5G は無線アクセス技術として物理層およびデータリンク層の要素を含み、TSN は主にデータリンク層に位置付けられる Ethernet 拡張技術ですが、同期や設定などの仕組みを含めると、より広いレイヤーに関連します。本書では、これらを「OPC UA が動作し得る下位ネットワーク技術の代表例」として簡略化して表現しています。

PubSub 通信についても、UDP や TCP を直接利用する形態に加え、MQTT や AMQP といったブローカ型プロトコルを介する構成など、複数のマッピングが存在します。図 8 では主要な要素のみを示していますが、すべての通信パターンやヘッダ構造を OSI 参照モデルに厳密に当てはめたものではありません。

以上のように、図 8 は OPC UA と下位通信技術の関係を理解するための概念的な整理図であり、特定のネットワーク技術や OSI 層への唯一の割り当てを示すものではありません。OPC UA は多様な通信環境やネットワーク構成と柔軟に組み合わせることができる点に特徴があり、この柔軟性こそが異種環境間での接続性を支える重要な要素となっています。

## 2.2　なぜ OPC UA が必要なのか？

製造現場やインフラ分野では、システムの多様化・ネットワーク接続範囲の拡大に伴い「安全に」「意味を保ったまま」、すなわちセキュリティを前提とし、データの意味（セマンティクス）を共有できる形で、データを相互接続する共通基盤が求められています。従来、プラントや工場などのコントローラ制御で相互運用を行う際には、機器ごとにプラットフォームが異なるため、独自にプログラムを作成する必要があり、多大な労力を費やしてきました。このような課題に対する初期の解決策の一つとして、OPC（OPC Classic）が誕生しました。

OPC Classic は、マイクロソフト社の COM/DCOM 技術を利用しており、Windows 環境で動作することを前提に作られたプロセス通信規格です。これにより、異なるメーカーや機器間でデータ送受信を行うことができました。OPC Classic は世界中の企業の製品に使用され、多くの産業分野で広く利用され、ICS 分野における代表的な通信技術の一つとなっていきました。

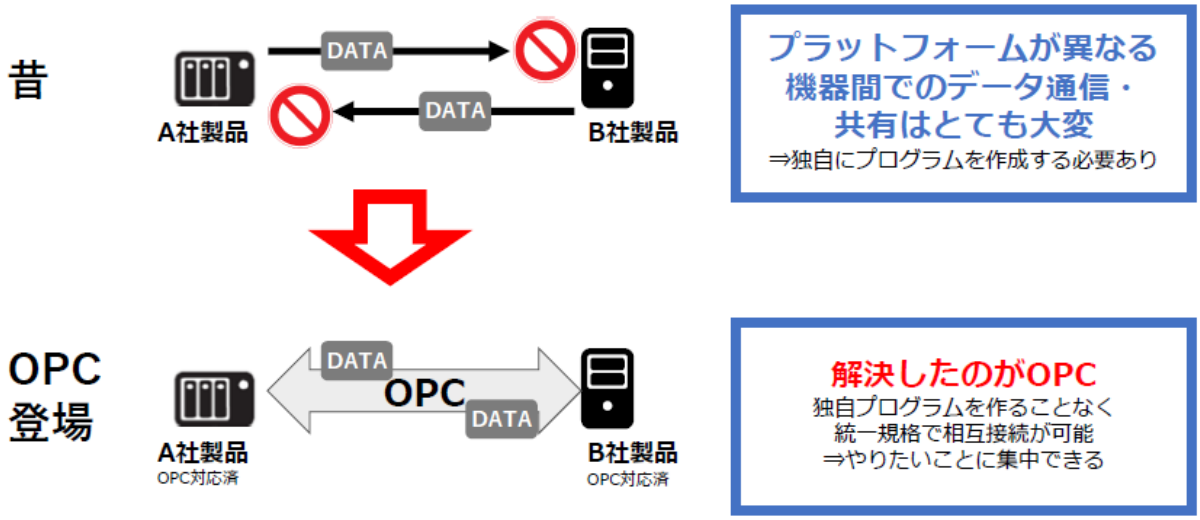



**図 9　従来の通信と OPC の通信の比較**

しかし、技術の進化に伴い、OPC Classic に対しても新たな要求が高まっていきました。例えば、以下のような要求があります。

- さらにオープンな環境向けの仕様
  OPC Classic は Windows をベースにした規格であるため、適用範囲に制約が出てしまいました。そのため、Windows に縛られない仕様が求められるようになりました。

- 常に進歩する脅威に対するセキュリティ要求
  OPC Classic が採用している通信プロトコルでは、ファイアウォールを介した通信を適切に行うことが難しく、また情報系ネットワーク上で利用する際に求められるセキュリティ機能も十分ではありませんでした。データの堅牢性や信頼性を保証する仕組みが不足していたのです。

- アプリケーションの高機能化
  単純な数値やメモリデータだけではなく、構造化されたデータや意味を持たせた情報をやり取りしたいという要望が出てきました。

このような市場の要求に応えるため、OPC UA が誕生しました。

OPC UA は、特定の通信方式やプラットフォームに依存せず、セキュリティを前提とした設計と、データの意味を共有できる情報モデルを中核に据えた点に特徴があります。また、OPC Classic の持つ優位性を継承しつつ、セキュリティとデータ交換性を飛躍的に向上させた拡張性の高い、標準化されたプラ

ットフォーム非依存の次世代 OPC 通信規格です。この特長によって、次のようなことが実現できます。

・適用範囲の拡大
　OPC UA はプラットフォームに依存しないため、Windows や Linux など、どんな環境でも動かせます。そのため、製造現場の機器からエッジ、クラウドまで幅広く利用できます。

・セキュリティが担保された通信
　証明書や暗号化といった標準的なセキュリティ手段を採用しているため、製造現場のデータを安全にクラウドに送信できます。

・意味のあるデータ交換
　情報や機能を表す「情報モデル」という統一されたモデリング機能を提供しています。これにより、交換したい情報を分類・意味づけして、テンプレートやモジュールとして表現できるようになりました。

　以上のように、OPC UA は単なる通信規格ではなく、セキュリティを前提とした信頼性の高い通信と意味のあるデータ交換（情報モデル）を両立する共通基盤として、現代の産業システムに求められる要件に応える技術であり、製造現場の機器レベルから上位のエッジやクラウドまでを一つの統一されたインタフェースで接続し、相互接続性・相互運用性を実現できます。

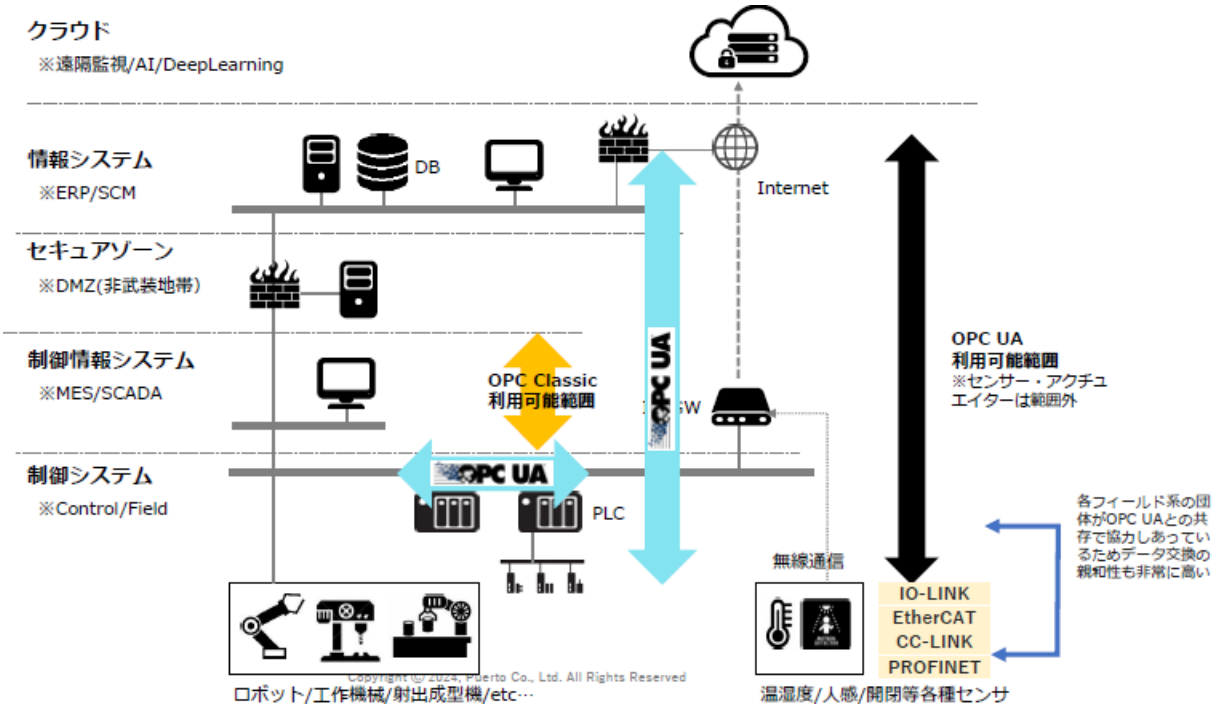


**図 10　OPC Classic の利用可能範囲と OPC UA の利用可能範囲**

　なお、図 10 は、PLC 間（コントローラ間）の通信については、PubSub を利用することも可能ですが、制御用途における相互運用性や構成管理の観点から、現在は OPC UA FX の利用が主に想定されています。

　こうした背景を踏まえて、次に既存技術との違いを整理します。

## 2.3　既存技術との通信の違い

OPC UA は、既存の通信技術と比較して、通信の考え方やデータの扱い方において多くの利点を持っています。ここでは、MQTT や HTTP/HTTPS（REST）といった代表的な通信技術と比較し、通信モデルや状態管理、データ表現の観点から OPC UA の強みを整理します。

### 2.3.1 OPC UA PubSub によるメッセージング連携

OPC UA は、Client/Server 通信に加えて PubSub（Publish/Subscribe）通信モデルを提供しており、メッセージング基盤と連携したデータ配信を実現できます。PubSub は、多数のデバイスから周期的にデータを配信する用途や、疎結合なシステム構成が求められる環境に適しています。

PubSub 通信の実装方式の一つとして、MQTT が広く利用されています。MQTT は軽量なプロトコルであり、ネットワーク帯域やリソースが限られた環境でも多数のデバイスから効率的にデータを収集することが可能です。

OPC UA over MQTT では、OPC UA の情報モデルが持つセマンティクスを維持したまま、MQTT のメッセージング基盤を利用できます。これにより、単なる数値データの転送にとどまらず、データの意味や構造を保持した形でクラウドや上位システムに配信することが可能となります。

なお、Client/Server 通信においては、OPC UA は TCP に加えて HTTPS や WebSocket を用いた通信もサポートしており、Web アプリケーションやクラウドサービスとの連携も可能です。これにより、用途に応じて PubSub と Client/Server を使い分けた柔軟なシステム構成が実現できます。

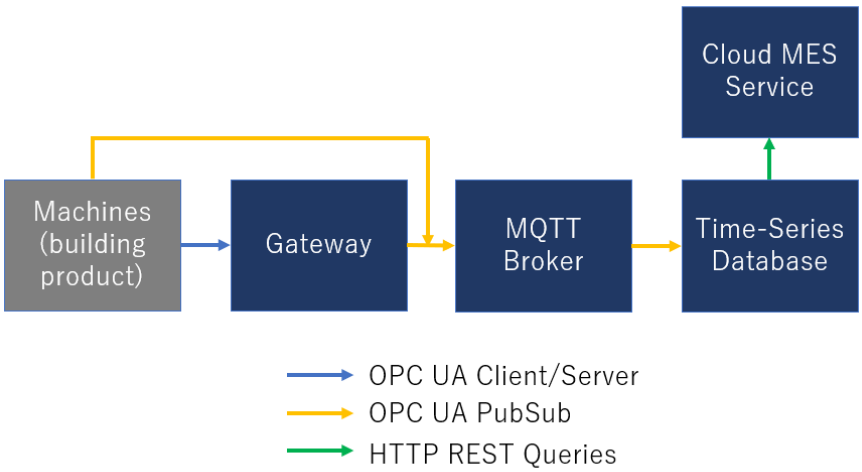



図 11　OPC UA over MQTT

### 2.3.2 REST との違い（ステートフル・ステートレス）

OPC UA（Client/Server）は、セッションを確立して状態を管理するステートフルな通信モデルとして設計されており、これによりサーバーは各セッションの状態性を追跡することができます。この特性により、ユーザーは過去の要求に基づいた情報を維持し、接続状態を保持したまま新たな要求を行うことが可能となります。

ここでいうステートフル／ステートレスは、REST（HTTP）と OPC UA の通信設計思想の違いを理解するための対比です。
OPC UA がステートレス動作を選択できるという意味ではなく、OPC UA（特に Client/Server 通信）はセッションを確立する設計である点が特徴です。

ステートフルな通信は、状態を考慮したデータの更新やイベント通知の受信を実現し、ユーザーに対してより柔軟で効率的なインタラクションを提供します。さらに、ユーザーは各セッションでデータの送受信を継続的に行うことができ、よりリアルタイムな情報交換が可能です。

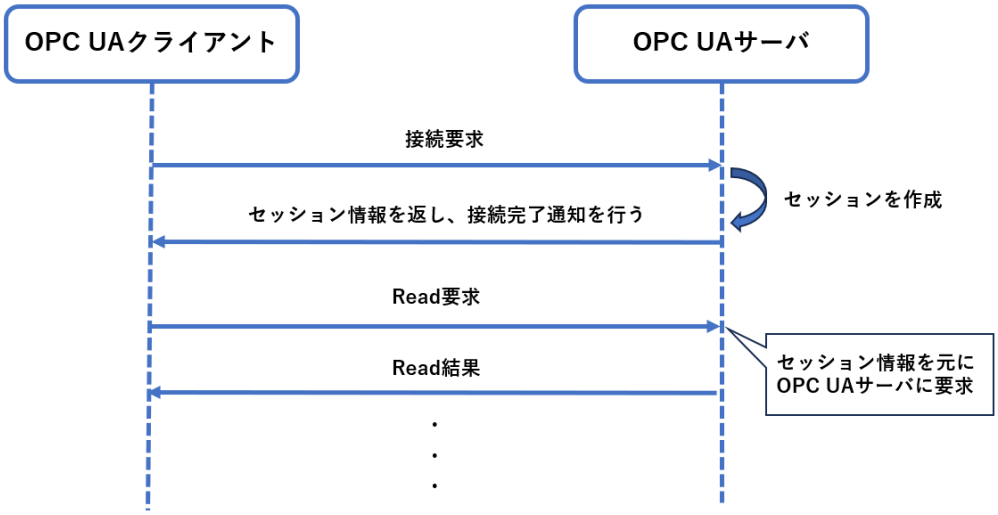



**図 12　OPC UA（Client/Server）のセッション概念**

一方で、REST/HTTP のようなステートレスな通信モデルは、過去のユーザーの要求に関する情報を保存しない特性を持っています。このアプローチでは、各要求が独立しており、相互に分離された形で処理されるため、サーバーは各リクエストに対して新たに必要な情報を提供する必要があります。

ステートレスな通信モデルは、システムのスケーラビリティや可用性を向上させる一方で、ユーザーが過去のコンテキストを考慮したインタラクションを行うことが難しくなる場合があります。

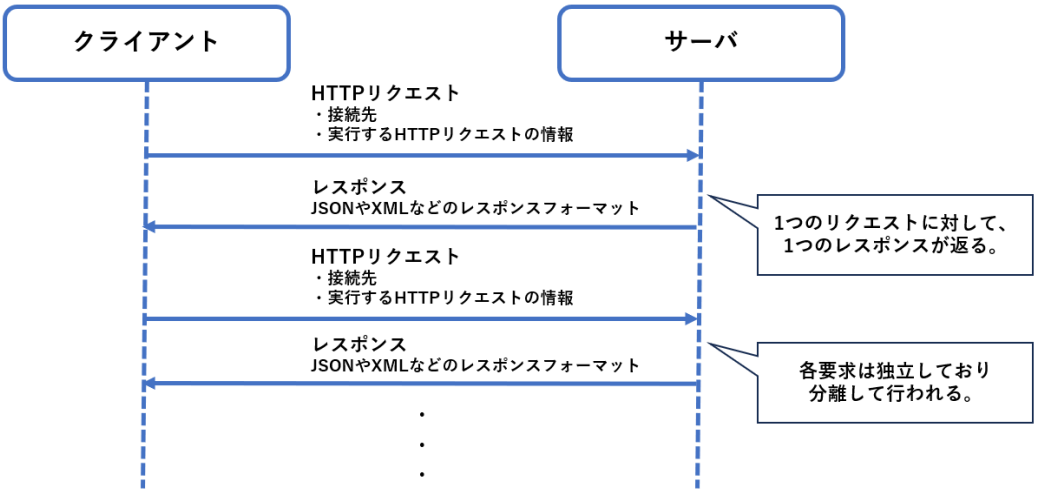



**図 13　REST/HTTP におけるステートレスな要求処理の概念**

ここまで通信モデルの違いを説明してきましたが、それらの設計思想は、アプリケーションの実装形態にも影響します。参考として、一般的なステートフルアプリケーションとステートレスアプリケーションの特徴を表 9 に記載します。

**表 9　ステートフルアプリケーションとステートレスアプリケーションの比較**

| 項目 | ステートフルアプリケーション | ステートレスアプリ |
|---|---|---|
| 状態の保持 | サーバーが状態（セッションなど）を保持する。 | 各リクエストが独立しており、状態を保持しない。 |
| リクエスト内容 | リクエストに基づいた内容。接続先などの一部の情報は省略可能。 | 接続先や実行する HTTP リクエストの情報をすべて含んだ内容。 |
| スケーラビリティ | サーバーごとにセッションの管理が必要であり、拡張が難しい。 | サーバー間で情報の連携がないため、拡張しやすい。 |

| 項目 | ステートフルアプリケーション | ステートレスアプリ |
|---|---|---|
| リソース使用量 | 状態などの情報をメモリに保存する必要があり、多い傾向がある。セッションを切断しないとリソースは解放されないことが多い。 | 状態を保存せず、リクエストごとに処理をするので、少ない傾向がある。 |
| ネットワーク負荷 | セッション作成後の負荷は低くなる傾向がある。ただし、セッションの同時接続数が増えると負荷が高くなる傾向がある。 | 接続先や実行する HTTP リクエストの情報を毎回送信する必要があるので負荷が高くなる傾向がある。 |

### 2.3.3 セマンティクス（意味表現力）の比較

OPC UA は、データをオブジェクトとして扱うことで、情報の意味（セマンティクス）を明確にし、効果的な情報のやり取りを実現します。各データは属性やメソッドを持つオブジェクトとして定義され、具体的な意味を持つ情報として扱われます。

このように情報モデルによってデータの意味と構造が標準化されることで、異なるメーカーやシステム間であっても共通の理解に基づくデータ連携が可能となり、相互運用性の向上と一貫したデータ活用が実現されます。

OPC UA といくつか通信方式におけるデータのセマンティクス（意味表現力）の比較を表 10 に記載します。

**表 10　セマンティクス（意味表現力）の比較**

| 通信方式 | 表現力 | 詳細 |
|---|---|---|
| HTTP | 低い | JSON や XML などで任意の構造を表現することができるが、構造に共通の規約がないため、データの意味的な表現力は低い。 |
| MQTT | 低い | トピックとペイロードのみでデータに意味を持たせることが難しいため、表現力は低い。 |
| MQTT Sparkplug | 高い | MQTT にメタデータや状態表現が追加されており、共通の規約でデータのやり取りを行うことができるため表現力は高い。 |
| OPC UA | 非常に高い | 情報モデル（データ型、構造体、オブジェクト、関係性、アクセス制御など）を定義でき、共通の規約でデータのやり取りを行うことができるため、表現力はかなり高い。 |

# 3　導入のメリットと簡単な事例

OPC UA は、産業機器や情報システムが扱う多様なデータを統一的に扱える通信基盤を提供することで、システム間の接続性と情報活用の効率性を高める役割を担います。

本章では、実際に導入された例や、導入によって得られるメリットについて、特に製造業および社会インフラ分野を中心に紹介します。
また、小規模環境からの導入手法や、初期検討時に有効なツール活用例についても触れ、具体的なイメージ形成を支援します。

## 3.1　製造業・インフラ業界での活用例

製造業や社会インフラの分野では、従来 PLC や DCS、SCADA などの機器・システムが多数導入されてきました。これらは一般にメーカーや世代が異なるため、相互接続やデータ統合の難しさが課題となっていました。OPC UA は、こうした異種システム間の共通言語として機能し、横断的なデータ活用を可能にしています。

■ 事例 1：生産ラインのリアルタイムモニタリング（製造業）
ある加工業企業では、複数ベンダー製の PLC を使用した生産ラインにおいて、各装置からのデータ収集を行うシステム構築に要する時間と手間がかかっていました。

OPC UA サーバー機能をもつゲートウェイを導入することで、各装置の稼働状態、アラーム情報、生産実績を統一フォーマットで収集できるようになり、ダッシュボードによるリアルタイム可視化とアラート通知の自動化が実現しました。

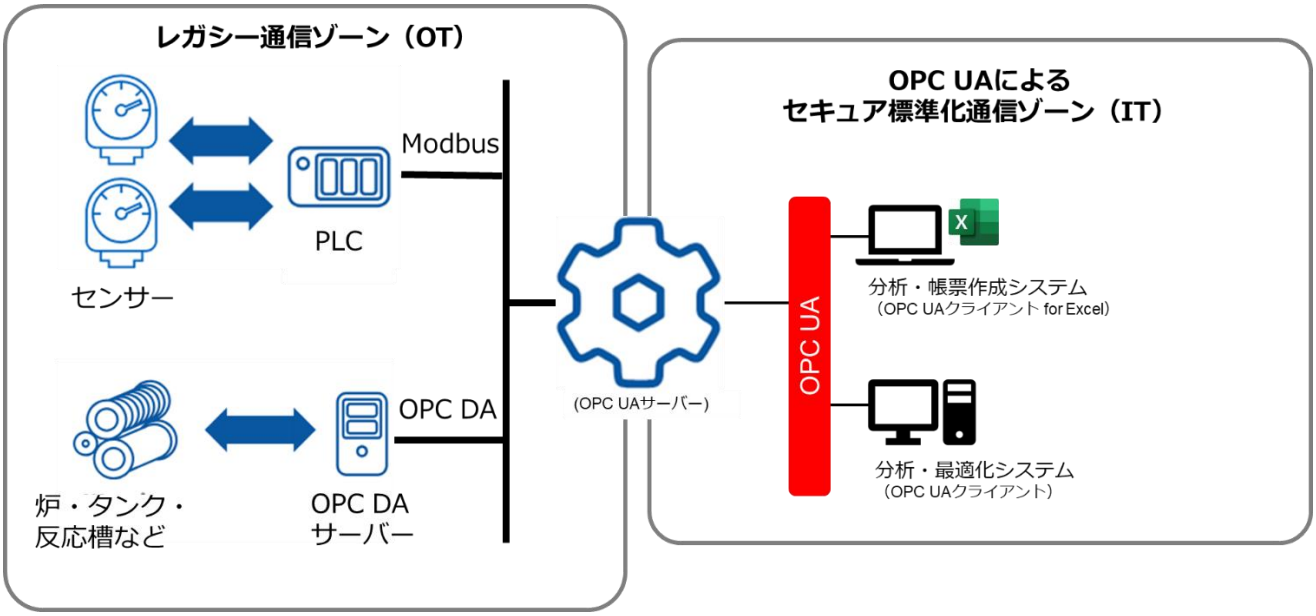



**図 14　生産ラインのリアルタイムモニタリングのイメージ図**

■ 事例 2：上下水処理設備の遠隔監視（インフラ業界）

ある自治体では、広域に分散した上下水処理設備を統合的に監視・管理する仕組みを構築するために、OPC UA を採用しました。既存の監視装置に OPC UA 変換機能を追加することで、拠点間の通信とデータ集約を統一プロトコルで実現し、遠隔拠点における設備の稼働状況や異常通知を本部側からリアルタイムで確認可能となりました。

これらの事例に共通するのは、**「既存設備を大きく変更せずに、段階的な接続と可視化を実現できた」**点にあります。

OPC UA はその柔軟性と拡張性から、小規模な PoC（概念実証）段階から全社的なデータ連携基盤への発展までを支援できる通信基盤です。

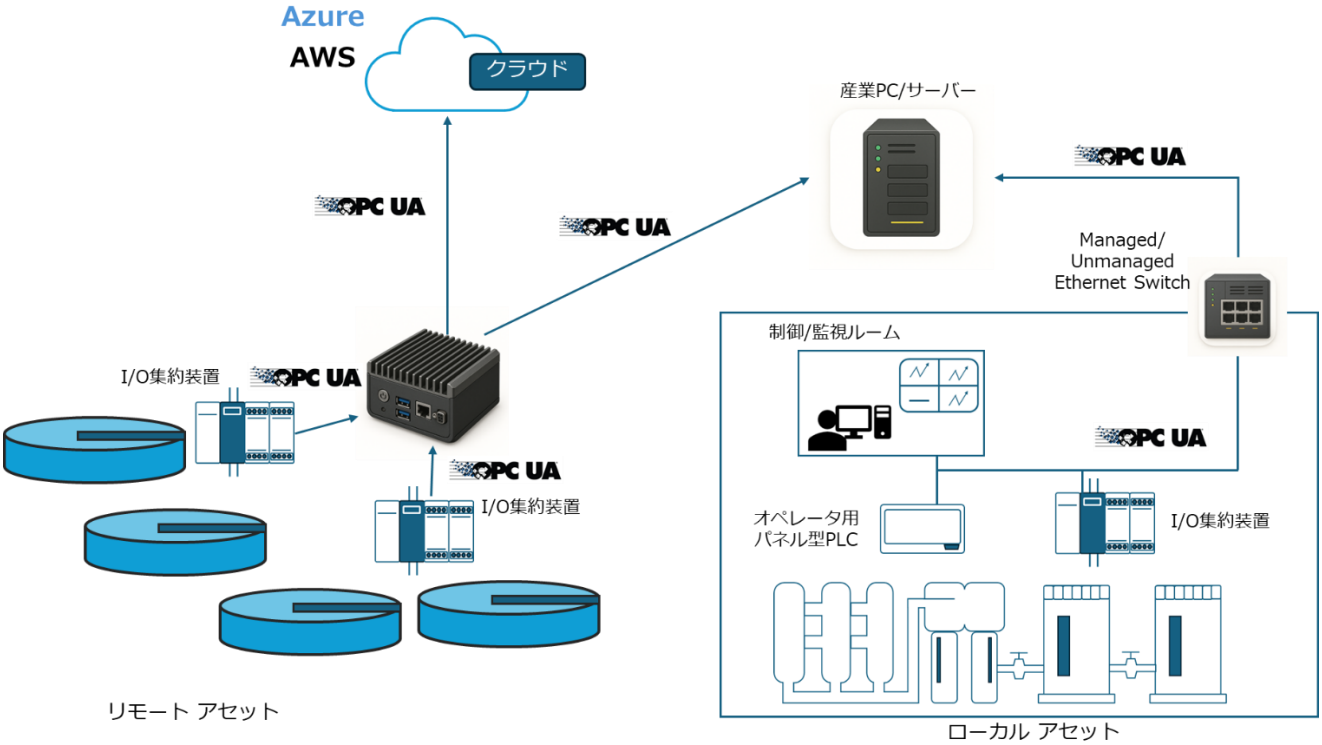


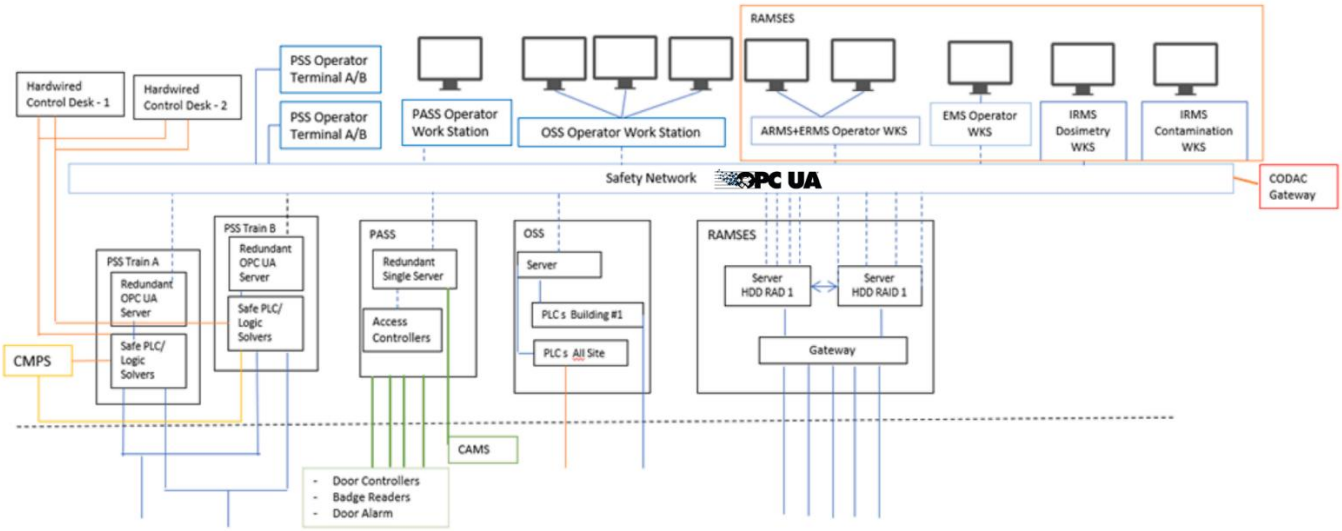

**図 15　上下水処理設備の遠隔監視イメージ**

■ 事例 3：核融合エネルギー施設における OPC UA を活用した次世代制御（インフラ業界）

将来のクリーンエネルギーを担うとされる核融合発電。その実現に向けた欧州の代表的な研究開発プロジェクトが「IFMIF-DONES（国際核融合材料照射施設）」です。この施設では、核融合炉に使用される材料を中性子で照射し、耐久性や安全性を評価するための試験が行われます。

こうした極めて高度かつ安全性が求められる環境において、次世代の中央制御システム（CICS）が導入されており、その中核を担うのが OPC UA を含む標準技術による分散制御アーキテクチャです。

IFMIF-DONES の制御システムは、中央から個々の装置までを統合管理する階層型の構成となっています。OPC UA は IFMIF-DONES において、特に安全関連制御（PSS）や装置制御における信頼性の高い通信インタフェースとして活用されています。




出典：「Recent advances of the IFMIF-DONES central instrumentation and control systems engineering design」

**図 16　核融合エネルギー施設における次世代制御イメージ**

## 3.2　導入による効果

OPC UA を導入することにより、設備やシステム間の接続性が向上するだけでなく、開発・保守の工数削減や業務の省力化・自動化といった多面的な効果が得られます。ここでは、実務における主な導入効果を 3 つの観点から整理します。

① データの標準化による開発効率の向上

OPC UA は、通信プロトコルだけでなく、データの意味構造（情報モデル）も標準化されています。そのため、複数ベンダーの機器を統一的な手法で扱えるようになり、アプリケーション開発やシステム設計の共通化・再利用性が高まります。

- ベンダーごとの個別ドライバ開発が不要
- インタフェース仕様のドキュメント化が容易
- 共通の API で複数システムへの対応が可能

特に OPC UA Companion Specification を活用した場合、各業界共通のタグ構造を元に開発できるため、プロジェクト間の設計整合性も高まります。

② 可視化・モニタリングによる業務の効率化

OPC UA は、クライアントアプリケーションやダッシュボードツールとの連携が容易であり、設備稼働のリアルタイム監視や異常検知の自動化に大きく貢献します。

- 各装置の稼働率、エラー履歴、トレンド分析の可視化
- データ取得周期や監視条件を柔軟に設定可能
- 異常時の通知・アラート連携が容易（メール、MQTT 等）

③ 保守・拡張時の柔軟性向上と省力化

OPC UA は、アドレス空間やノード構成を柔軟に定義できるため、システムの変更や拡張に対しても迅速に対応可能です。また、通信のセキュリティや認証も標準で備えているため、運用時の設定変更やメンテナンスも比較的容易です。

- センサ追加や設定変更に伴う変更作業を最小限に抑制
- セキュアな通信が既定仕様のため、追加開発が不要
- 新旧設備の混在にも対応しやすい構成が可能

OPC UA 導入により、「データを集めること」から「活用できる状態に整えること」への転換が進み、現場の見える化・省人化・効率化に向けた第一歩として機能します。

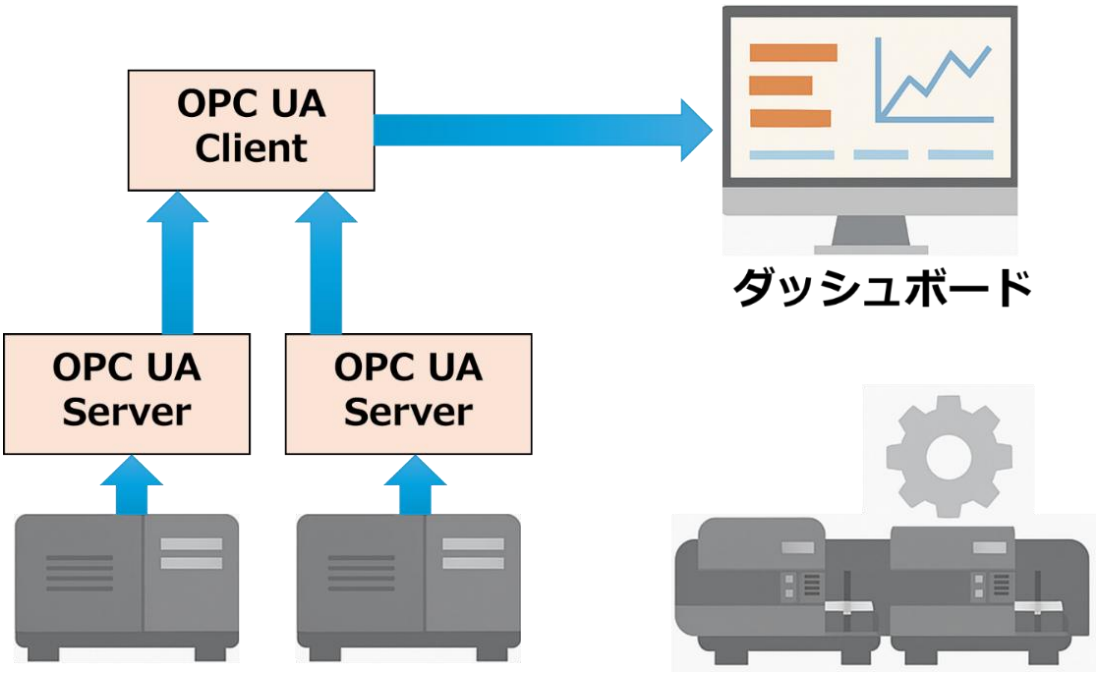



図 17　OPC UA による設備モニタリング構成例

## 3.3　小規模から始める方法

OPC UA はスケーラブルな通信基盤であり、小規模な PoC（概念実証）や試験導入から段階的に展開することが可能です。本節では、初めて OPC UA に取り組む際に実行しやすい方法と、導入の足がかりとなるツールやライブラリについて紹介します。

①オープンソースツール・SDK の活用

初期段階では、商用製品の導入に先立ち、オープンソースや無償提供されている公式 SDK を活用したプロトタイピングが有効です。以下のよく利用されているライブラリ・ツールの一覧を示します。

表 11　OPC UA ツール

| 名称 | 提供元 | 概要 | 主な用途 |
|---|---|---|---|
| UA-.NETStandard | OPC Foundation | C#/.NET 向けの公式 OPC UA SDK。機能・仕様ともに本仕様の標準実装に最も近く、教育・評価にも適する。 | Windows および Linux 環境におけるサーバー／クライアントアプリのプロトタイピングや社内ツール開発 |

UA-.NETStandard SDK は、OPC Foundation GitHub にて無償公開されており、学習・評価用途での活用に適しています。

② サンプル構成の例（1:1 通信の最小構成）

小規模導入の典型的な構成として、以下のような 1 台の OPC UA サーバー（アプリケーション）と 1 台のクライアント（ツール）による最小構成から試すことを推奨します。

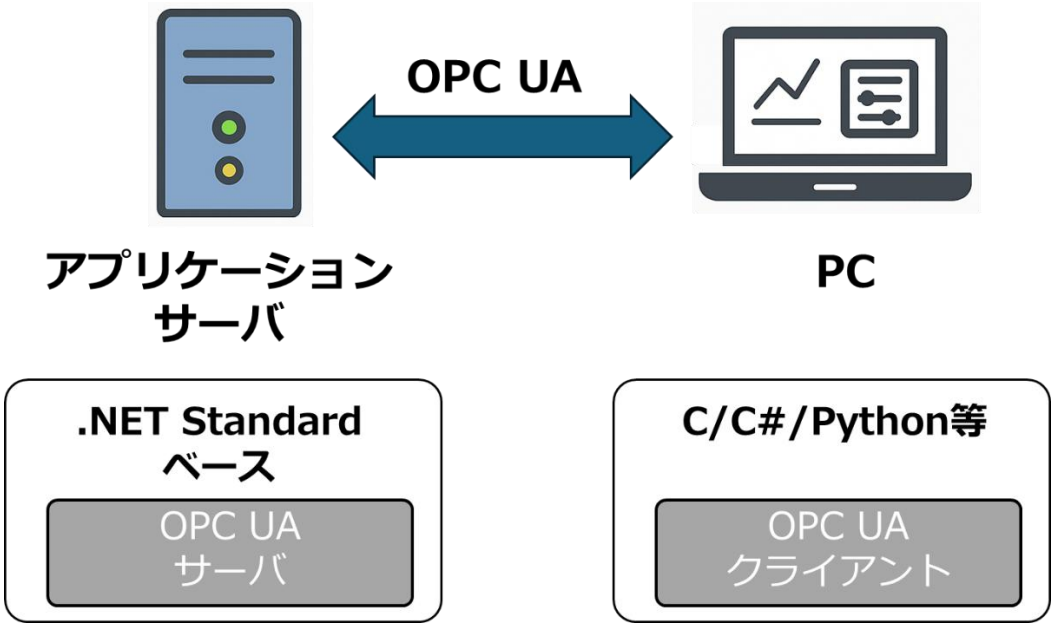



図 18　小規模 OPC UA 構成

③ 最初に試すシナリオ例

- サーバー側で「温度」「圧力」などのノードを定義し、シミュレーションデータを提供
- クライアントから値を読み取り、手動で書き込みテストを実施
- サーバー、クライアント間のセキュリティ設定の有効性確認

このプロセスを通じて、「通信の確立」「ノードモデルの理解」「セキュリティ設定の基本」など、実運用に向けた基礎スキルが身につきます。

④ 本格展開へのステップアップ

PoC 後は、以下のように段階的な拡張を行うことで、実業務への適用が現実的になります。

- 複数設備との同時接続やサブスクリプション活用
- 情報モデル（OPC UA Companion Specification 等）への対応
- セキュアな証明書運用の導入
- IT 側との接続：MES・クラウドサービスとの統合検証

このように、OPC UA は.NET ベースの公式 SDK を活用することで、Windows／Linux 問わず、無償で小規模導入を始められる柔軟性を持っています。また、Client/Server 通信を入口として導入を進める場合でも、将来的な拡張や高度な相互運用を見据えた指針として、OPC Foundation が公開している OPC 11030（OPC UA FX に関するガイドライン） などの資料を参照することで、段階的な発展を検討することが可能です。

# 4　導入時のポイント

OPC UA の導入にあたっては、単に通信機能を有効にするだけでなく、機器構成・データ設計・セキュリティ・開発体制など、複数の観点から計画的に進めることが重要です。特に既存設備やシステムとの接続を行う場合、段階的な展開と初期設計の妥当性が成功の鍵となります。

本章では、導入検討段階において注意すべき代表的なポイントを整理するとともに、よくある課題や誤解に対する対策についても触れ、現場での実装や運用に役立つ情報を提供します。

## 4.1　OPC UA の基本的な導入プロセス

OPC UA の導入は、単なる通信機能の有効化ではなく、設備の可視化・制御・連携を意識した構成設計とデータ設計が不可欠です。ここでは、初期段階で検討すべき導入プロセスを、主な 7 ステップに整理して紹介します。

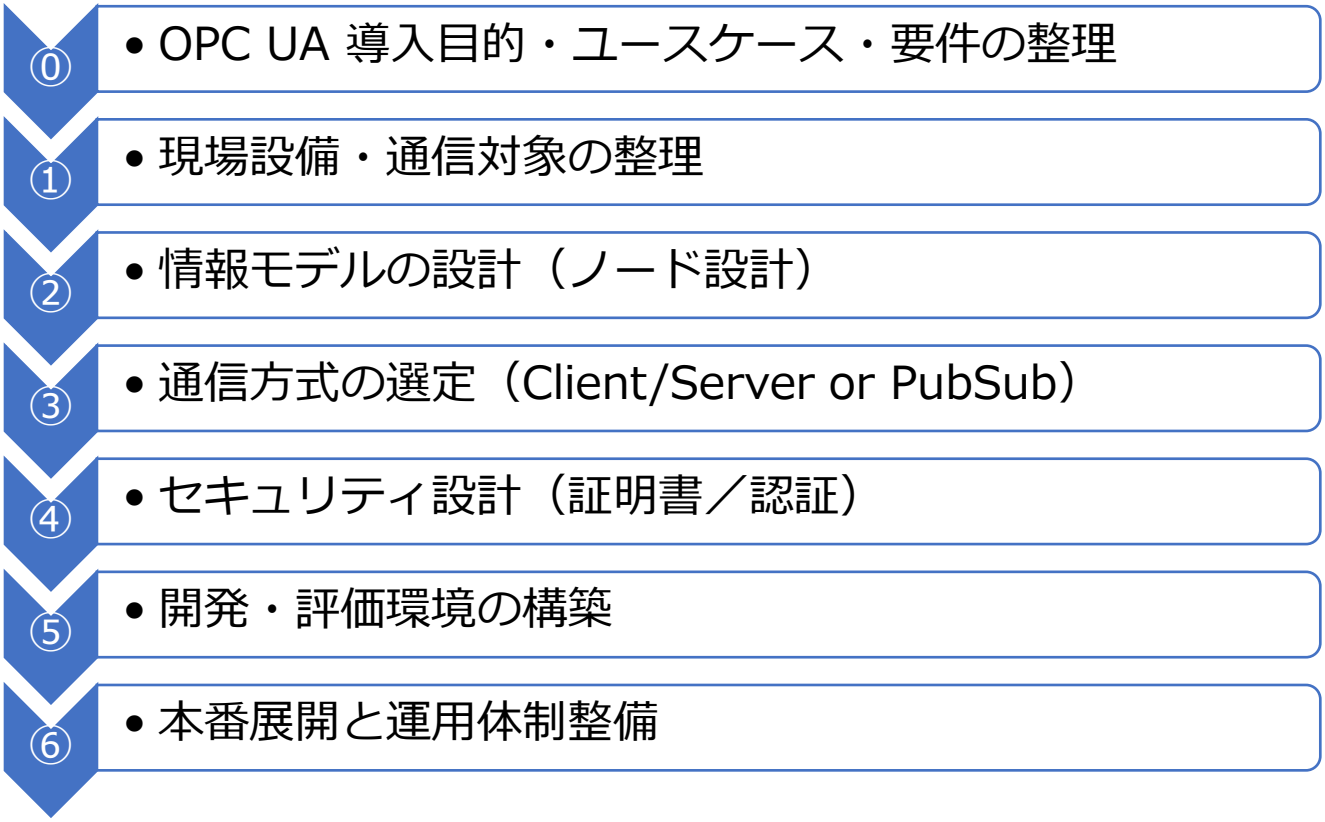



**図 19　OPC UA の基本的な導入プロセス**

⓪ OPC UA 導入目的・ユースケース・要件の整理

まずは、OPC UA を導入して「何を実現したいのか（ユースケース）」を明確にし、関係者間で合意形成します。目的と要件が曖昧なまま情報モデルや通信方式の検討に進むと、取得すべきデータ項目・粒度・取得周期や、必要なセキュリティレベルが過不足となり、後工程での手戻りにつながります。ここで整理した内容は、以降の情報モデル設計（どの情報をどう表現するか）、通信方式の選定（Client/Server／PubSub、更新周期やイベント通知の要否）、セキュリティ設計（認証・権限・監査）、運用設計（証明書更新や障害対応）に直接影響します。

① 現場設備・通信対象の整理

OPC UA 通信の対象となる現場の設備・センサ・制御機器の棚卸しを行い、接続対象の構成と通信要件を明確にします。

- どの機器と通信する必要があるか？
- 通信対象は読み取りのみか、書き込みも必要か？
- 既存の PLC や装置が OPC UA をサポートしているか？

既存機器が OPC UA 非対応の場合は、OPC UA ゲートウェイの活用やラッパーによる変換も検討されます。

② 情報モデルの設計（ノード設計）

OPC UA は、単なる値の読み書きによるデータ通信ではなく、「情報構造（モデル）」を定義することで意味のあるデータ交換を行います。まず情報モデルとして型（Type）を設計し、その型をもとに実際の設備や装置に対応するノードをインスタンスとして生成・配置することで、アドレス空間が構成されます。

ここでは、取得すべきデータ項目を「ノード」としてどのように整理・定義し、モデル設計とアドレス空間設計の両面から検討すべきポイントを示します。

- 変数名、単位、アクセス権限の定義
- センサ・装置ごとのツリー構造設計
- OPC UA Companion Specification に準拠するか、独自モデルとするか
  （Companion Specification では、NodeSet2.xml として情報モデル定義ファイルが公開されており、これを利用・拡張することが可能）

③ 通信方式の選定（Client/Server or PubSub）

使用目的やシステム規模に応じて、適切な通信モデルを選択します。

**表 12　OPC UA の通信モデル一覧**

| 通信モデル | 主な用途 | 特徴 |
|---|---|---|
| Client/Server | 小～中規模、双方向制御向け | 安定性・双方向通信・認証の柔軟性が高い |
| PubSub (Publish/Subscribe) | 大規模／リアルタイム性重視 | 軽量・一方向配信に適する、疎結合な構成 |

④ セキュリティ設計（証明書／認証）

OPC UA では、通信モデルに応じたセキュリティ機構が定義されています。

Client/Server 通信では、証明書ベースの相互認証と暗号化（SecureChannel）と、ユーザー認証／アクセス制御が標準で提供されます。一方、PubSub 通信では、メッセージ指向の通信モデルに適した暗号化および認証方式が規定されています。

導入時には以下を検討します。

- 通信ポリシー（セキュリティモード、証明書ペアの管理）
- ユーザー認証（ユーザー名＋パスワード、証明書、匿名許可など）
- アクセス制御（ロール定義、読み書き権限設定）

⑤ 開発・評価環境の構築

PoC や検証目的で、まずは評価環境を構築します。

- 推奨 SDK（例：UA-.NETStandard）や評価ツールを使用
- 仮想環境・Raspberry Pi 等で試作・実証
- （実装依存）診断ログや通信トレースを有効化し、トラブルシューティングに備える

⑥ 本番展開と運用体制整備

評価を経て有効性が確認できた場合、本番環境への展開を検討します。
この際、OPC UA を安全かつ安定的に運用するため、以下の点について整理・整備が必要です。

- OPC UA 通信を前提としたネットワーク設計
  （IP アドレス設計、通信経路、ファイアウォールでのポート／通信方向の整理）

- 証明書および信頼関係の運用管理
  （証明書の配布・更新・失効管理、信頼ストアの管理方針）

- OPC UA サーバー／クライアント障害時の運用フロー整備
  （接続断・証明書エラー等を想定した対応手順、運用部門との連携）

このように、OPC UA の導入は「技術仕様の理解」だけでなく、「システム設計・セキュリティ・運用までを一体で考える」アプローチが求められます。

## 4.2　よくある課題と解決策

OPC UA の導入においては、仕様理解・通信構成・セキュリティ・運用設計など、さまざまな観点で課題に直面するケースがあります。本節では、現場でよく寄せられる質問をカテゴリ別に整理し、それぞれの背景と推奨される対応策を FAQ 形式で解説します。

【通信・接続編】

Q1　「OPC UA クライアントがサーバーに接続できません。何を確認すべきですか？」

A　以下の点を順に確認してください：

1. サーバー側で対象ポート（通常 4840 番など）が開いているか（ファイアウォール含む）
2. サーバーの証明書がクライアントに信頼されているか（クライアント側の信頼リストに登録）
3. クライアントの証明書がサーバーに信頼されているか（サーバー側の信頼リストに登録）
4. IP アドレス／ホスト名の設定が正しいか
5. クライアントとサーバーで、エンドポイント設定（SecurityPolicy、MessageSecurityMode）が一致しているか

OPC UA クライアントなどで「ログ」を確認すると、接続失敗の詳細原因が特定しやすくなります。

Q2　「通信が遅く感じられる／タイムアウトするのはなぜですか？」

A　通信レイテンシの主な要因は以下です：

- ポーリング通信による周期制限（Client/Server モデルでの過剰な読み取り）
- ネットワークの混雑や VPN 越しの帯域制限
- サーバーのサブスクリプション処理負荷が高い
- PubSub 構成での QoS やメッセージサイズの設定ミス

対策としては、サブスクリプション型通信の活用、送信頻度の見直し、PubSub の適用検討などが有効です。

【機器・互換性編】

Q3　「OPC UA に対応していない既存設備はどうすれば良いですか？」

A　既存機器が OPC UA 非対応であっても、以下の手段があります：

- ゲートウェイ装置（Modbus, Ethernet/IP などを OPC UA に変換）
- ラッパーソフトウェア（既存 DLL や API を OPC UA ラッパーで公開）
- 自社製 OPC UA サーバーの開発（推奨 SDK の活用）

通信プロトコル変換や、ソフトウェア設計による中間層の導入が鍵となります。

【セキュリティ編】

Q4　「証明書の扱いが難しいのですが、どう運用すればよいですか？」

A　OPC UA では、通信の暗号化および通信相手の正当性を保証するために、証明書ベースの相互認証と暗号化（SecureChannel）が標準で採用されています。そのため、OPC UA を運用する際には、証明書の生成・配布・信頼管理といった証明書運用が不可欠な要素となります。

基本的な運用方法は以下のとおりです：

- 自己署名証明書の生成と交換（初期段階）
- 信頼済み／拒否済み証明書のリスト管理
- 本番環境では企業 CA による証明書署名の導入が望ましい

評価環境では「証明書の自動承認」設定も可能ですが、本番ではセキュリティリスクとなります。

Q5　「ユーザーごとのアクセス制限はできますか？」

A　はい、OPC UA は「Role-based Access Control（RBAC）」によってユーザー単位のアクセス権限を制御できます。

- ユーザーごとに「読み取り専用」「書き込み可能」「管理者」などの役割を設定
- ノードごとにアクセス権限を設定可能（RolePermissions）

セキュリティポリシーとユーザー管理の整備が重要です。ただし、サポートしているかどうかは製品提供ベンダーにご確認ください。

【開発・拡張編】

Q6　「今は PoC だけど、本番展開に耐えられる構成にできますか？」

A　OPC UA はもともとエンタープライズ～フィールドレベルまでの拡張性を前提に設計されており、以下のようなステップアップが可能です：

- 小規模環境 → 複数拠点連携、PubSub やクラウド統合へ
- OSS ベースの構成 → 商用 SDK やツールを併用した冗長構成へ
- 単純なデータ公開 → 情報モデルを活用したシステム統合へ

これらの課題は、導入前に全てを解決する必要はありません。段階的に対応範囲を広げ、実務と並行して理解を深めることが、導入成功への近道です。

# APPENDIX 用語一覧

| カテゴリ | 用語 | 説明 |
|---|---|---|
| 規格・名称 | OPC | OPC（OLE for Process Control の略、現在は Open Platform Communications と再定義）。 |
| 規格・名称 | OPC UA | OPC Unified Architecture の略。産業通信の標準規格。 |
| 規格・名称 | OPC Classic | 従来の OPC 規格。OPC DA, OPC HDA などを含む。 |
| 規格・名称 | COM/DCOM | Microsoft が提供する分散オブジェクト技術。 |
| 規格・名称 | ICS | Industrial Control Systems（産業制御システム）の略。 |
| 通信モデル | Client/Server モデル | クライアントが要求し、サーバーが応答する従来型の通信モデル。 |
| 通信モデル | PubSub モデル | Publish/Subscribe の略。非同期で効率的なデータ配信を行う通信モデル。 |
| 通信モデル | OPC UA PubSub | OPC UA が規定する Publish/Subscribe 通信仕様。 |
| 情報モデル | アドレス空間 | OPC UA サーバー内に実際に存在するノードの集合。 |
| 情報モデル | 情報モデル | ノードや関係をどのように設計・定義するかを示す枠組み。 |
| 情報モデル | セマンティクス | データが「何を表しているのか」という意味や文脈を、機械が解釈できる形で表現する考え方。<br>OPC UA では、情報モデル（ノード、メソッド、属性、参照）によってセマンティクスを実装します。 |
| 情報モデル | Node（ノード） | OPC UA の基本要素。Variable, Method, Event などを含む。 |
| 情報モデル | Variable（変数） | ノードの状態や値を表す要素。 |
| 情報モデル | Method（メソッド） | ノードに紐づく操作や処理を表す要素。 |
| 情報モデル | Event（イベント） | システム内で発生する事象を表す要素。 |
| 情報モデル | Attribute（属性） | ノードの特性やメタデータを示す情報。 |
| 情報モデル | Reference（参照） | ノード間の関係を定義するリンク。 |
| セキュリティ | TLS | Transport Layer Security。暗号化通信の標準規格。 |
| セキュリティ | PKI | Public Key Infrastructure。公開鍵基盤。 |
| セキュリティ | Role-based Access Control（RBAC） | ユーザーの役割に基づいてアクセス制御を行う仕組み。 |
| セキュリティ | 証明書 | 通信や認証に利用されるデジタル証明書。 |
| ネットワーク | TCP/IP | インターネットやローカルネットワークで利用される通信プロトコル群。 |
| ネットワーク | HTTP / HTTPS | Web 通信に利用される標準プロトコル。 |
| ネットワーク | WebSocket | 双方向通信を可能にするプロトコル。 |
| ネットワーク | JSON | 軽量なデータ記述形式。 |
| ネットワーク | XML | 階層構造を持つデータ記述形式。 |
| ネットワーク | バイナリ | 機械が処理する 0/1 ベースのデータ形式。 |
| ネットワーク | SDK | Software Development Kit。開発用のツール群。 |