**OPC (Open Platform Communications)**

OPC is implemented in server/client pairs.

The OPC server is a software program that converts the hardware communication protocol used by a PLC into the OPC protocol.

The OPC client software is any program that needs to connect to the hardware, such as an HMI.

The OPC client uses the OPC server to get data from or send commands to the hardware.

<https://www.opcdatahub.com/WhatIsOPC.html#:~:text=OPC%20is%20a%20software%20interface,PLC%20into%20the%20OPC%20protocol>.

**How does OPC work?**

OPC specification describes the *interface between clients and servers, servers and servers, including access to real-time data, monitoring of alarms and events*, access to historical data and other applications.

* The value of OPC is that it is an open standard, which means **lower costs for manufacturers and more options for users**.
* Hardware manufacturers need only provide a **single OPC server** for their devices to communicate with any OPC client.
* Software vendors simply include OPC client capabilities in their products and they become instantly compatible with thousands of hardware devices.
* Users can choose any OPC client software they need, resting assured that it will communicate seamlessly with their OPC-enabled hardware, and vice-versa.

**OPC Aggregation:**

Connect an OPC client to several OPC servers.

**OPC Tunneling**:

Connect an OPC client to an OPC server over a network.

**OPC Bridging:**

Connect an OPC server to another OPC server to share data.

--------------------------------------------------------------------------

The OPC DataHub (Data – Integration) is uniquely designed to do all of these tasks. It is a combination OPC server and OPC client that supports multiple connections.

Thus it can connect to several OPC servers simultaneously, for OPC aggregation and OPC bridging.

Two OPC DataHubs can mirror data across a TCP network to provide OPC tunneling.

**In addition to enhancing OPC server and client connections, the OPC DataHub can connect any OPC server or client to other applications as well, such as**[**Excel**](https://www.opcdatahub.com/Features/OPC_to_Excel.html)**, a**[**web browser**](https://www.opcdatahub.com/Features/OPC_to_Web.html)**, or any**[**ODBC database**](https://www.opcdatahub.com/Features/OPC_Logging.html)**. And it can also be used to get OPC data into**[**Linux or QNX**](https://www.opcdatahub.com/Features/OPC_to_LinuxQNX.html)**.**

Following OPC specifications are used in industries:

* **OPC-DA (Data Access):** Provides access to real-time data. We can query most recent values of temperature, pressure, density, acceleration, and other types of process control data from OPC-DA server.
* **OPC-HDA (Historical Data Access):** This is used to retrieve historical process data for analysis. This data is typically stored in files, databases or remote telemetry systems.
* **OPC-AE (Alarms & Events):** OPC AE servers are used to accept and exchange process alarms and events.
* **OPC-DX (Data exchange):** It defines the way OPC server data is exchanged with other OPC servers.
* **OPC-XML (XML Data Access):** It defines schema and data representation formats based upon XML standard. This makes it possible to share & manipulate process control data across all operating systems – Windows, UNIX, Solaris, etc.

### Benefits of using OPC standard are:

* Reduced load on the hardware device.
* Increased scalability of the system.
* Because of OPC server, client applications need not know anything about hardware protocol details.
* Though device need not serve multiple clients, So Increased life for the device.
* Interoperability (Unix/Linux and Windows – both platforms are supported by OPC)
* Standardization

**OPC – UA (Unified Architecture):**

The OPC Unified Architecture (UA), released in 2008, is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework.

The UA Design Specification goals are:

1. **Functional** **Equivalence**: All COM (Component Object Mode) OPC Classic Specification mapped to UA.
2. **Secure**: Encryption, Authentication and auditing.
3. **Extensible**: Ability to add new features without affecting old features.
4. **Comprehensive information modeling:** for defining complex information

* The OPC UA information modeling framework turns data into information. With complete object-oriented capabilities, even the most complex multi-level structures can be modeled and extended.
* This framework is THE fundamental element of OPC Unified Architecture. It defines the rules and base building blocks necessary to expose an information model with OPC UA.
* While OPC UA already defines several core models that can be applied in many industries, other organizations build their models upon them, exposing their more specific information with OPC UA.

OPC-UA is the successor to OPC-DA (Data Access). It has many benefits, one of the most prominent being that it is platform agnostic. There are SDKs available in a multitude of languages and OPC-UA can be implemented on anything from a tiny embedded to device running Linux, to a desktop running Windows, or a server running the OS of your choice.

**Data Logging:**

**Store process data in your database**

Modern process integration often calls for factory data to be stored in a database, such as Access, SQL Server, Oracle or MySQL. The OPC DataHub provides a unique configuration interface that makes this task both quick and completely open to customization.

UA ---> If the Client is in one machine and server is in another machine then you can able to communicate successfully using End point(EX: opc.tcp://IpAddress/Desktopname:27730(port number)).

DA----> Server and Client must be in one machine to communicate each other.

### BECK OPC UA:

### The Kolibri library allows the implementation of a producer client application for the com.tom

### PORTAL Services. It implements the Kolibri protocol for communication with the com.tom

PORTAL broker component and offers an API for configuration as well as data transmission.

**Besides the data exchange between the application and the library, which is initiated from the**

**application, the library carries out the following activities in an internal task:**

* Establish a WebSocket connection to the broker (HTTP upgrade mechanism)
* Execution of the Kolibri protocol
* Logging in to the broker
* Query of the broker existing nodes
* Data transmission from the registered data record with respect to the configured trigger

mode and Quality-of-Service level

* Data receiving for the registered data record in consideration of the configured Quality-of-

Service-Levels.

* Synchronization of data communication with the broker and with the application.

**Suppose manaku 3 devices(PLC, motor, sensor) unnay. Then manaku HMI, Database store, machine condition moniter unnay bcs manam chusko danki and then control cheydanki…**

**Without opc prathi device ki oka driver add cheyali. So manaku 3 drivers kavali for every device connection ki.. so total ga 9 kavali…**

**Bcs of this there is over burden on the devices while transferring the data.**

**To reduce and to make it scalable we are adding opc on the top of devices. So with one opc driver we can manage everything like hmi, datastore, machine control etcc…**