Automation
For Future Industry

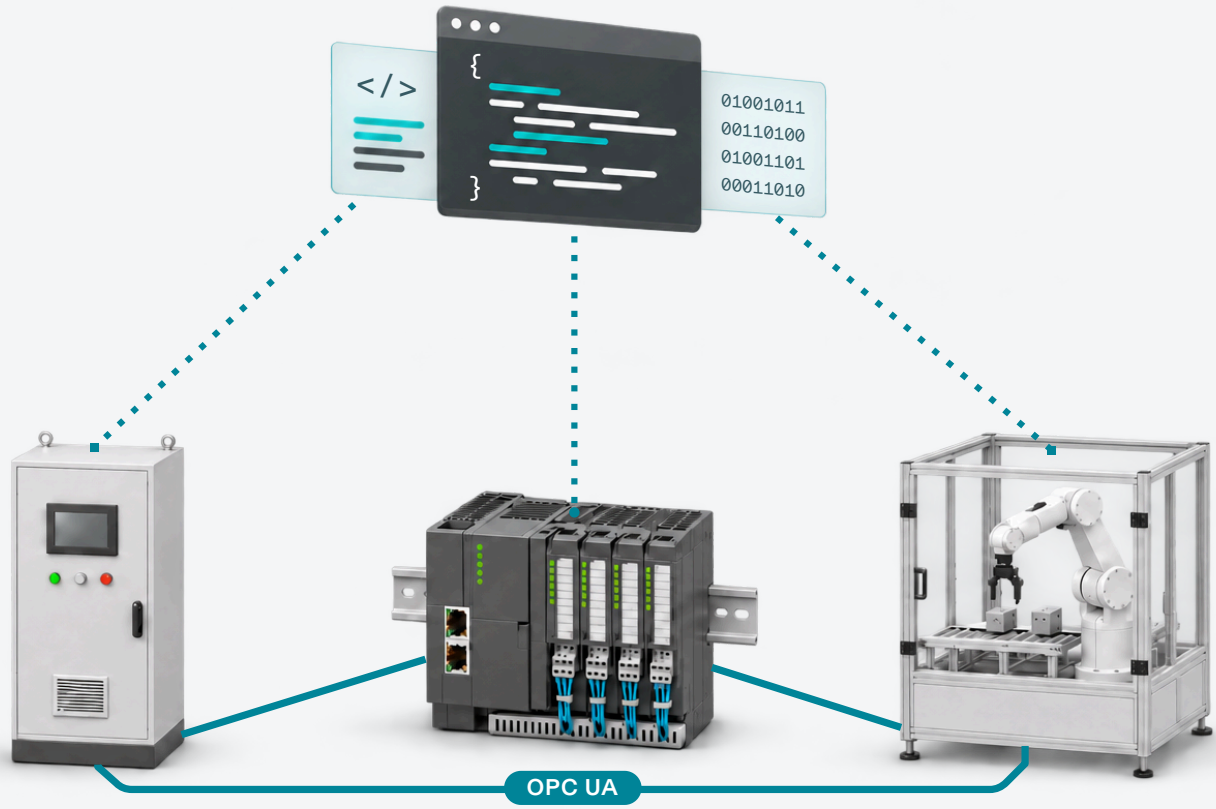
PRODUCT OVERVIEW
Version 2

open62541 | o6\Python | o6\C++ | o6\Rust

OPC UA SDKs

Professional OPC UA connectivity. Certification-ready solutions.

Build industrial OPC UA applications in C, Python, C++ and Rust.
One certified foundation, from first prototype to production.



PRODUCTION READY
Proven industrial core.

CRA READY
Lifecycle support available.

CERTIFICATION READY
Build on certified foundations.

WHY BUILD ON open62541?

Established. Fast. Trusted in industry.

We started open62541 and lead its development with a global community. It is the shared foundation for o6\Python, o6\C++ and o6\Rust.

open62541

13+ years

Continuous development since 2013, from embedded devices to industrial systems.

Quality

Continuous integration, automated testing and continuous code analysis.

Widely adopted

Integrated in industrial products. Open source under MPL 2.0.

Security and readiness across the SDK family.

Vulnerability management

Private reporting, assessment, validated fixes and coordinated disclosure through o6 PSIRT.

CRA readiness

Support for your CRA process: SBOMs, lifecycle commitments and maintenance agreements.

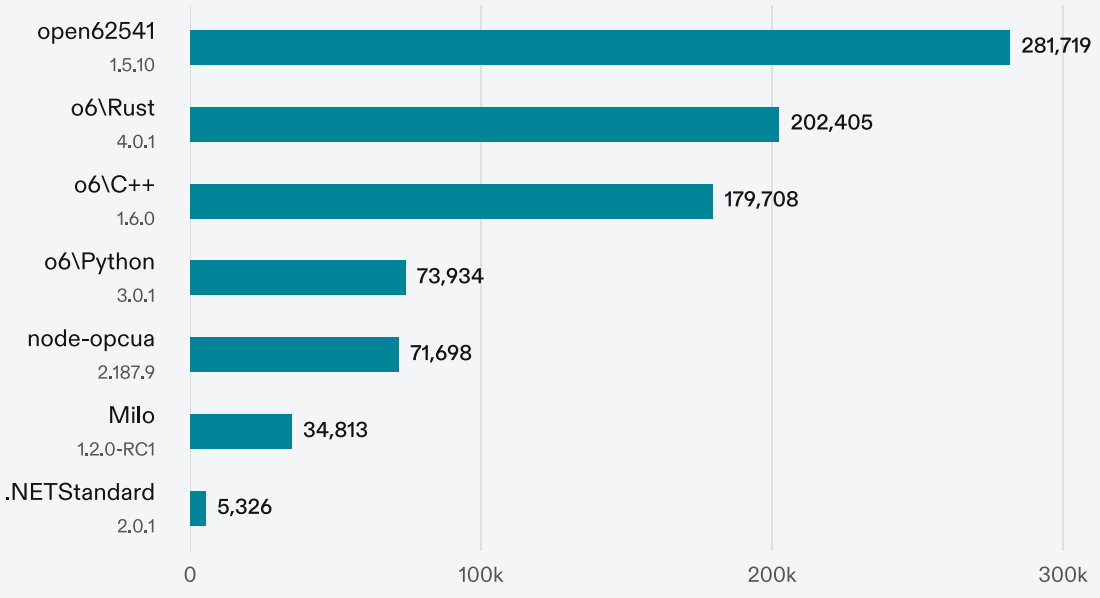
Certified foundation

Each of the SDKs achieved OPC Foundation certification for the Standard Server Profile.

OPC UA Server Throughput.

AVERAGE REQUESTS / SECOND

Unencrypted single-value reads | Higher is better



Recorded test system: Intel Core i5-1345U, Linux (WSL2).

Details and further information: github.com/o6-automation/opcua-benchmarks

THE o6 SDK FAMILY

One foundation. Your language. Your choice.

o6\Python

Embedded namespaces
Companion specs as native Python namespaces.

Automatic object mapping
Typed objects, fields and browse-name access.

Easy to use
Create nodes and read data with a concise Python API.

MCP integration
Connect AI assistants with the supplied MCP bridge.

AI enabled
NumPy data for analytics, AI and computer vision.

Native performance
OPC UA processing runs in the open62541 core.

o6\C++

Native app integration
Qt, Asio and native application event loops.

Explicit ownership
RAII value types manage native resource lifetimes.

Industrial toolchains
C++14+, CMake and direct native C access.

o6\Rust

Safe ownership
Typed values and resources through a safe Rust API.

Async & blocking APIs
Async with Tokio support, plus a blocking API.

Companion specifications
Bundled models, enabled through Cargo features.

Transparent Pricing.

Open source	Client	Client + server + PubSub
AGPL	€450	€950
Research & private use. AGPL terms apply.	Per developer, per year.	Per developer, per year.

GET STARTED WITH OPC UA

Solutions for your application. Start with a few lines.

open62541 — server.c

```c
UA_Server *s = UA_Server_new();
UA_Double v = 22.5;
UA_VariableAttributes a =
    UA_VariableAttributes_default;
a.displayName=UA_LOCALIZEDTEXT("","Temperature");
UA_Variant_setScalar(&a.value, &v,
    &UA_TYPES[UA_TYPES_DOUBLE]);
UA_Server_addVariableNode(s, UA_NODEID_NULL,
    UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER),
    UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES),
    UA_QUALIFIEDNAME(1, "Temperature"),
    UA_NODEID_NUMERIC(0, 63), a, NULL, NULL);
UA_Server_runUntilInterrupt(s);
```

o6\Python — server.py

```python
import o6

server = o6.Server(port=4840)
server.addVariable(
    "Temperature", server.objects, 22.5
)

with server:
    input("Press Enter to stop.\n")
```

o6\C++ — server.cpp

```cpp
#include <o6/server.hpp>

int main() {
    o6::Server server;
    server.addVariableScalar(
        "Temperature", "", 22.5
    );
    server.runUntilInterrupt();
}
```

o6\Rust — server.rs

```rust
let mut server =
    Server::new(&ServerConfig::default())?;
server.add_variable(
    NewVariable::new(
        NodeId::numeric(1, 1001), "Temperature"
    ).value(22.5_f64),
)?;
server.run_until(|| false)?;
```

Get more info

Explore the documentation and connect with our team.

o6-automation.com/o6-sdk

o6 Automation
For Future Industry

o6 Automation GmbH
Rintheimer Straße 23
76131 Karlsruhe, Germany

o6-automation.com
contact@o6-automation.com

OPC FOUNDATION MEMBER