Mittelstand-Digital Zentrum Magdeburg

Mittelstand-Digital

Gefördert durch:
Bundesministerium für Wirtschaft und Klimaschutz
aufgrund eines Beschlusses des Deutschen Bundestages

WORKSHOP: OPC UA AUF BASIS VON PYTHON
02.03.2023

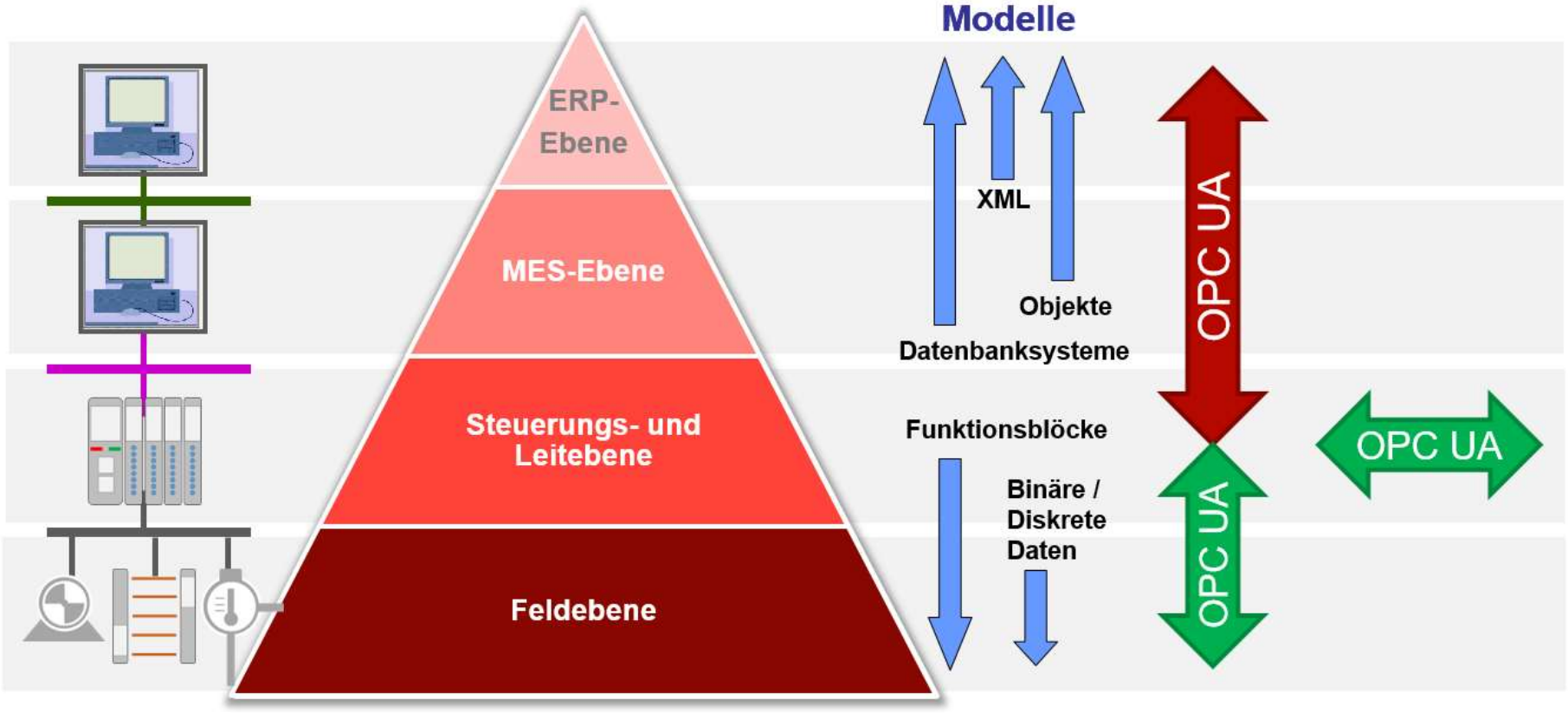
Mittelstand Digital Zentrum Magdeburg

Mario Thron (mario.thron@ifak.eu), Dr. Matthias Riedl (matthias.riedl@ifak.eu)

ifak Institut für Automation und Kommunikation e.V.
Werner-Heisenberg-Str. 1
39106 Magdeburg
Tel.: +49 391 990140
www.ifak.eu

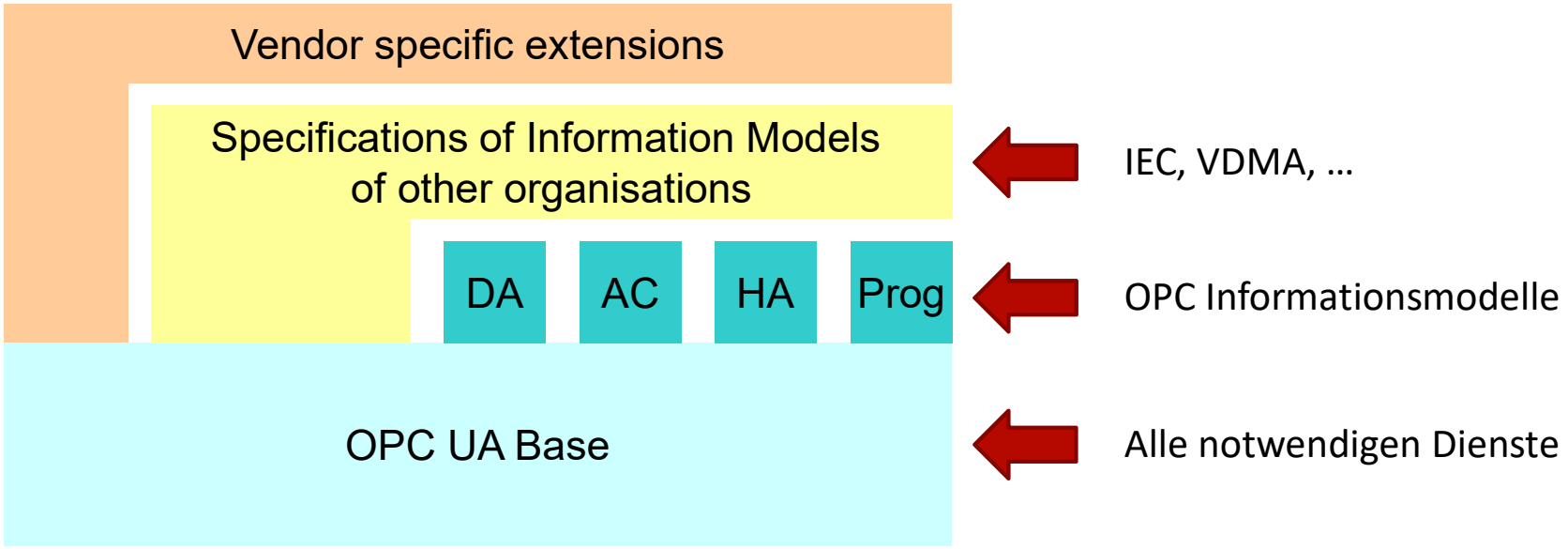
Agenda

Warum sollte man sich mit OPC UA beschäftigen?



Warum sollte man sich mit OPC UA beschäftigen?

Standardisierungsgremien und Industrieverbände erarbeiten gemeinsam mit den Herstellern und Anwendern von Kommunikationslöungen Spezifikationen zum Informationsaustausch auf Basis von OPC UA.



Warum sollte man sich mit OPC UA beschäftigen?

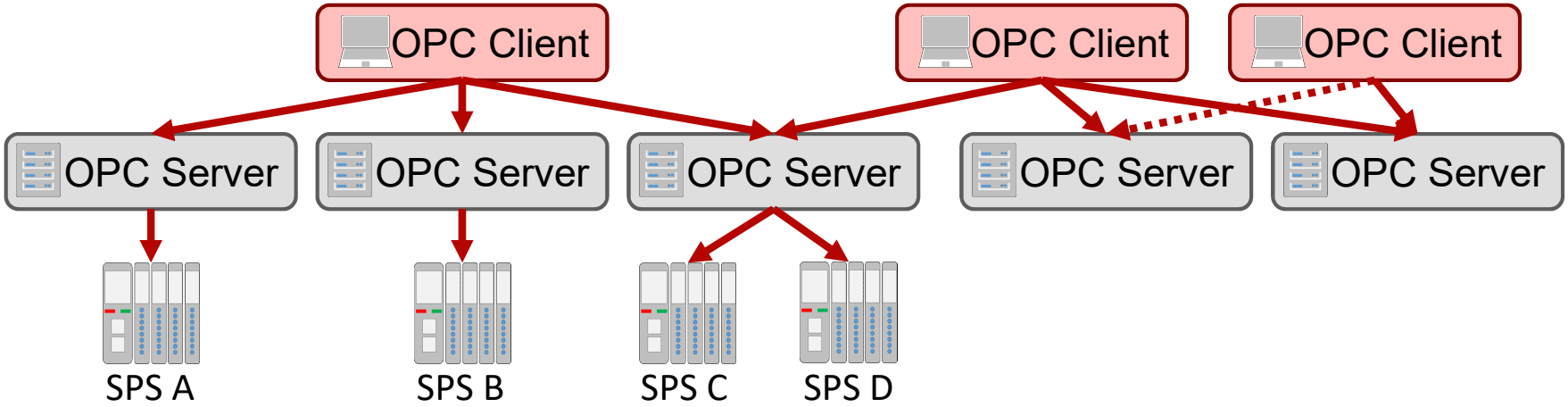
- Informations-Modelle, von Industrie-Gruppen erarbeitet und spezifiziert
- Gleichartige Maschinen / Systeme (Typen) enthalten gleichartige Datenmodelle
- Beispiele:
  - Roboter
  - Krane
  - Spritzgussmaschinen
  - Steuerungen, generisch

Agenda

Was ist eigentlich OPC UA?

- OPC - Open Platform Communications
  (vor 2011: OLE for Process Control, mit OLE - Obect Linking and Embedding)
- UA - Unified Architekture - Plattformunabhängie Kommunikation

- Client – Server Architektur



Was ist eigentlich OPC UA

- OPC UA Spezifikationen
  - Part 1 – Concepts
  - Part 2 – Security Modell
  - Part 3 – Address Space Model
  - *Part 4 – Services*
  - Part 5 – Information Model
  - Part 6 – Service Mappings
  - Part 7 – Profiles
  - *Part 8 – Data Access*
  - Part 9 – Alarm and Conditions
  - Part 10 – Programs
  - Part 11 – Historical Access
  - Part 12 – Discovery
  - Part 13 – Aggregates
  - Part 14 – PubSub
  - Part 15 – Safety
- OPC Foundation: OPC UA Reference: https://reference.opcfoundation.org
- Unified Automation, Viele Infos und C++ Stack: http://documentation.unified-automation.com/uasdkcpp/1.7.3/html/index.html

Agenda

Wie programmiert man OPC UA Anwendungen?

- Notwendige Arbeitsschritte
  - Installation von Python
  - Entwicklungsumgebung (?)
  - Installation notwendiger Softwarebibliotheken
  - Installation eines generischen OPC UA Clients
  - 3 Beispiele: Programmierung mehrerer Server und Clients in Python

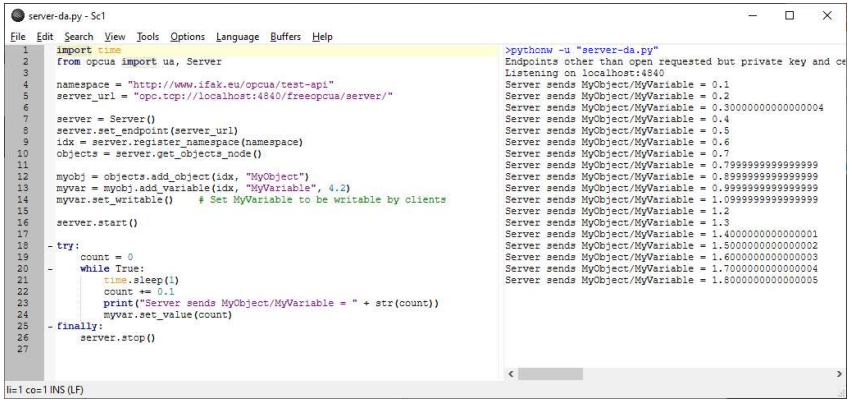
Installation von Python

- Warum?
  - Weite Verbreitung, Studenten haben oft Grundkenntnisse
  - Funktioniert auf Kleinstrechnern, Desktoprechnern, Cloud-Rechnern
  - Einfach zu nutzen (Interpreter)
- Was?
  - Python Foundation: https://www.python.org/
  - Download: https://www.python.org/downloads/ (wir nutzten Version 3.10)
  - Dokumentation: https://docs.python.org/3.10/
- Wie?
  - Windows: Download und Ausführung der *Installationsdatei*
  - Linux (Ubuntu, Debian, Raspbian): *sudo apt install python3*

Entwicklungsumgebung (?)

SciTE (einfacher Editor, Open Source)

https://www.scintilla.org/SciTEDownload.html
Programmstart: F5

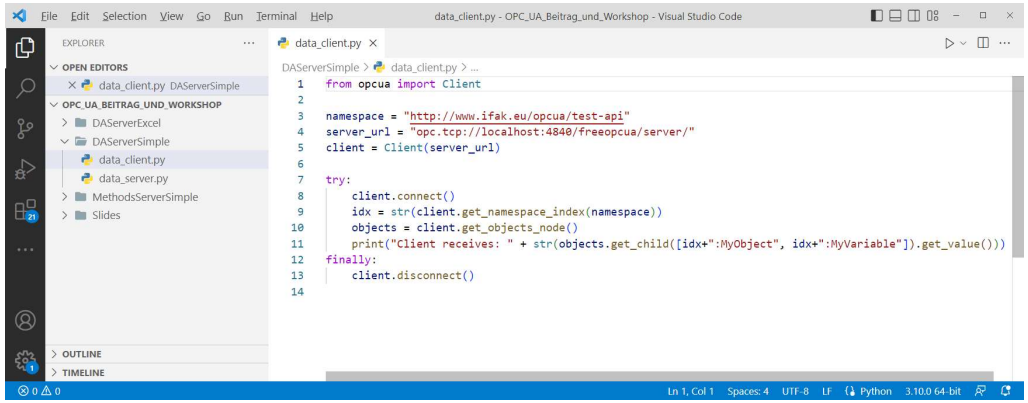


Visual Studio Code (IDE, Microsoft, Open Source)

https://vscodium.com/ oder https://code.visualstudio.com/
Programmstart: F5

Microsoft's `vscode` source code is open source (MIT-licensed), but the product available for download (Visual Studio Code) is licensed under this not-FLOSS license and contains telemetry/tracking. According to this comment from a Visual Studio Code maintainer: …

The VSCodium project exists so that you don't have to download+build from source. This project includes special build scripts that clone Microsoft's vscode repo, run the build commands, and upload the resulting binaries for you to GitHub releases. **These binaries are licensed under the MIT license. Telemetry is disabled.**

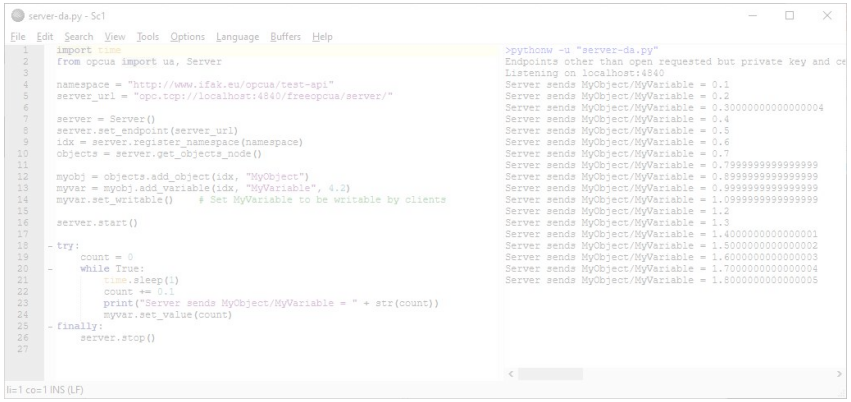


Entwicklungsumgebung (?)

SciTE (einfacher Editor, Open Source)

https://www.scintilla.org/SciTEDownload.html

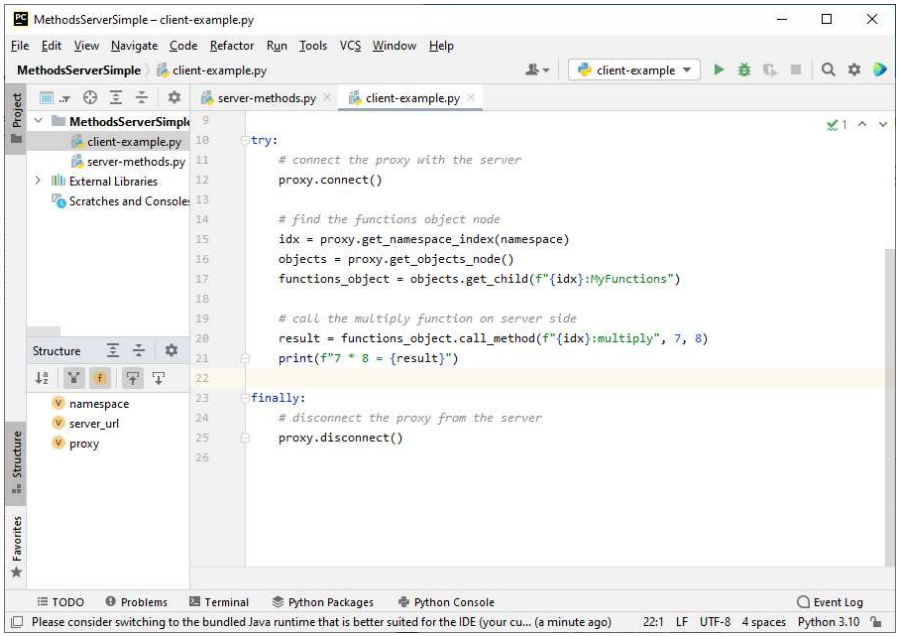
Programmstart: F5



PyCharm (alternative vollständige IDE)

https://www.jetbrains.com/de-de/pycharm/

Programmstart: Shift-F10



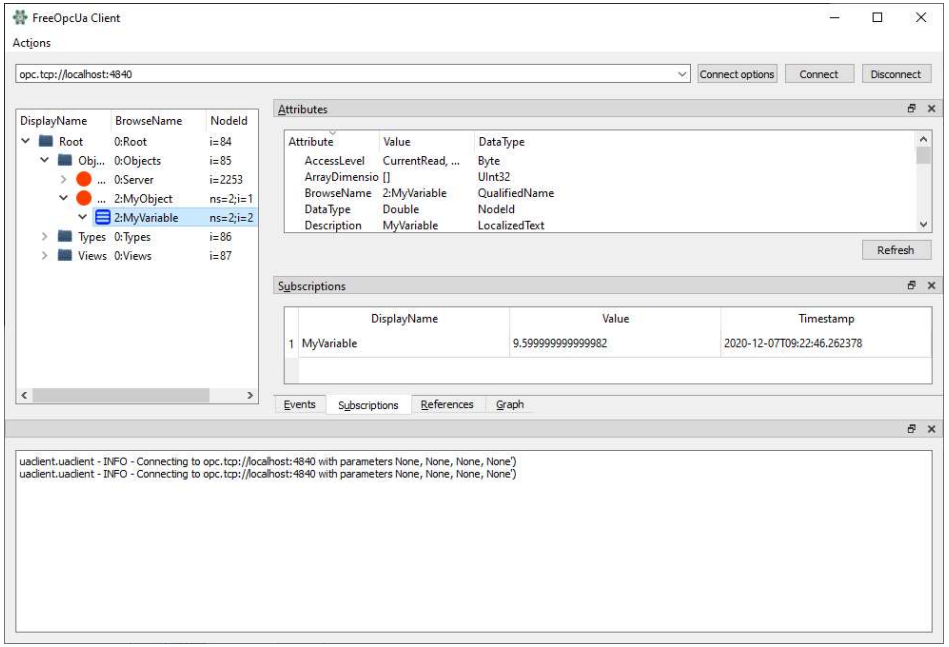
Installation notwendiger Softwarebibliotheken

- Paketverwaltungsprogramm: *pip* | *pip3* (kommt mit Python mit)
- Server von dem die Quellen kommen: https://pypi.org/

- Download der OPC UA Bibliothek
  python -m pip install --user opcua
- Informationen über diese Bibliothek
  - Freier Quellcode: https://github.com/FreeOpcUa
  - Dokumentation: https://python-opcua.readthedocs.io/en/latest/
  - Beispiele: https://github.com/FreeOpcUa/python-opcua/tree/master/examples

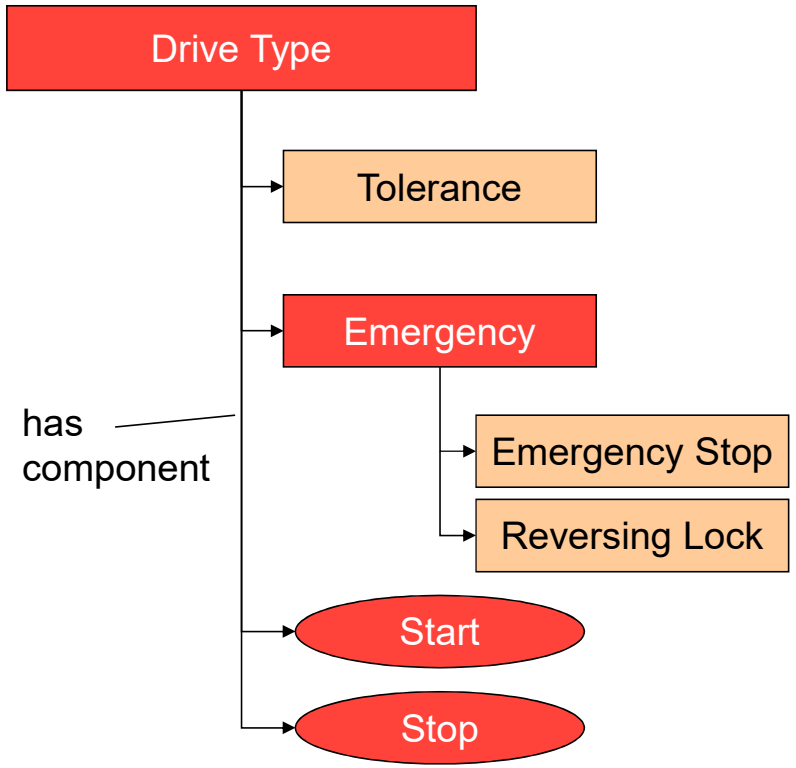
Installation eines generischen OPC UA Clients

- Installation via pip:
  python -m pip install --user opcua-client
- Starten des Clients
  opcua-client
- Standard-Adresse
  opc.tcp://localhost:4840

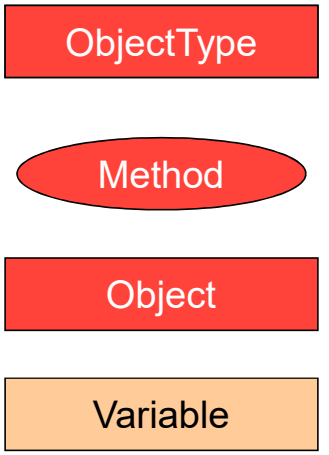


- Umfangreichere Alternativen:
  - https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-client/
  - https://www.unified-automation.com/products/development-tools/uaexpert.html

OPC UA Knotentypen



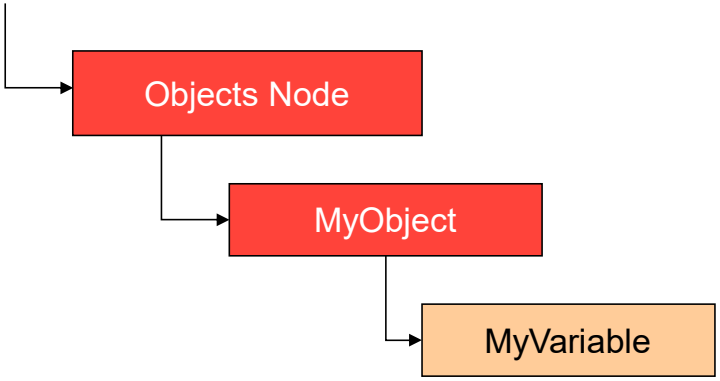
Legende:



Data Access: Server

```python
import time
from opcua import ua, Server

namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"

server = Server()
server.set_endpoint(server_url)
idx = server.register_namespace(namespace)
objects = server.get_objects_node()

myobj = objects.add_object(idx, "MyObject")
myvar = myobj.add_variable(idx, "MyVariable", 4.2)
myvar.set_writable()    # Set MyVariable to be writable by clients

server.start()

try:
    count = 0
    while True:
        time.sleep(1)
        count += 0.1
        print(f"Server updates MyObject/MyVariable = {str(count)}")
        myvar.set_value(count)
finally:
    server.stop()
```

OPC UA Server: *opc.tcp://localhost:4840/freeopcua/server/*



Data Access: Client

```
from opcua import Client

namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"
client = Client(server_url)

try:
    client.connect()
    idx = str(client.get_namespace_index(namespace))
    objects = client.get_objects_node()
    print("Client receives: " + str(objects.get_child([idx+":MyObject", idx+":MyVariable"]).get_value()))
finally:
    client.disconnect()
```

Methodenaufruf: Server

```python
from opcua import ua, uamethod, Server
import time

# configuration parameters of the server
namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"  # open for everybody: "0.0.0.0:4840"

# here is the function, which we want to publish over OPC UA
@uamethod
def multiply(parent, x, y):
    print("multiply method call with parameters: ", x, y)
    return x * y

# instantiation of the server and basic configuration
server = Server()
server.set_endpoint(server_url)
server.set_server_name("FreeOpcUa Method Server")
idx = server.register_namespace(namespace)

# get Objects node, this is where we should put our custom stuff
objects = server.get_objects_node()
my_functions = objects.add_object(idx, "MyFunctions")

# prepare in/out variables for multiply
x, x.Name, x.DataType = ua.Argument(), "x", ua.NodeId(ua.ObjectIds.Int64)
y, y.Name, y.DataType = ua.Argument(), "y", ua.NodeId(ua.ObjectIds.Int64)
result, result.Name, result.DataType = ua.Argument(), "Result", ua.NodeId(ua.ObjectIds.Int64)
my_functions.add_method(idx, "multiply", multiply, [x, y], [result])

try:
    # start the server
    server.start()
    print("Multiply OPC UA server started ...")

    # here follows a trick in order to wait for keyboard interrupt CTRL-C
    while True:
        time.sleep(3)

finally:
    # in all cases stop the server
    server.stop()
    print("Multiply OPC UA server stopped.")
```

Methodenaufruf: Client

```python
from opcua import Client

# configuration parameters of the server
namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"

# instantiate the client
proxy = Client(server_url)

try:
    # connect the proxy with the server
    proxy.connect()

    # find the functions object node
    idx = proxy.get_namespace_index(namespace)
    objects = proxy.get_objects_node()
    functions_object = objects.get_child(f"{idx}:MyFunctions")

    # call the multiply function on server side
    result = functions_object.call_method(f"{idx}:multiply", 7, 8)
    print(f"7 * 8 = {result}")

finally:
    # disconnect the proxy from the server
    proxy.disconnect()
```

Excel-Tabelle mit OPC UA Daten befüllen: Server

```python
import random
import time
from opcua import ua, Server

# configuration parameters of the server
namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"

# instantiation of the server and basic configuration
server = Server()
server.set_endpoint(server_url)
idx = server.register_namespace(namespace)
objects = server.get_objects_node()

# define variables
myobj = objects.add_object(idx, "Machine")
up_time = myobj.add_variable(idx, "UpTime", 4.2)
up_time.set_writable()
product_counter = myobj.add_variable(idx, "ProductCounter", 4.2)
product_counter.set_writable()

# start the OPC UA server
server.start()

try:
    # initialize the process variables
    t_start = time.time()
    product_counter_value = 0

    # continuously update variables
    while True:
        # sleep one second ...
        time.sleep(1)

        # update the process variable values
        up_time_value = time.time() - t_start
        product_counter_value += 10 + random.randint(0, 3)

        # message on server side
        print(f"Process variables:  up-time = {up_time_value:.2f} seconds,  total producted products = {product_counter_value}")

        # update the values in the server address space
        up_time.set_value(up_time_value)
        product_counter.set_value(product_counter_value)

finally:
    # in any case: stop the server
    server.stop()
```

Excel-Tabelle mit OPC UA Daten befüllen: Client

```python
import time
import win32com.client as win32
from opcua import Client
import os


# configuration parameters of the server
namespace = "http://www.ifak.eu/opcua/test-api"
server_url = "opc.tcp://localhost:4840/freeopcua/server/"

# init Excel and load a predefined table
excel_application = win32.Dispatch('Excel.Application')
filepath = os.path.join(os.getcwd(), 'ProcessMgmt.xlsx')
excel_file = excel_application.Workbooks.Open(filepath)
excel_sheet = excel_file.ActiveSheet
excel_application.Visible = True
row = 6  # start with data entries in row 6

# instantiate the client accessing the server
proxy = Client(server_url)
try:
    # connect to the server and create a reference to the objects node
    proxy.connect()
    idx = str(proxy.get_namespace_index(namespace))
    objects = proxy.get_objects_node()

    # continuously update some cells in Excel
    while True:
        time.sleep(2)  # every 2 seconds ...

        # ... update the index in the Excel table
        excel_sheet.Cells(row, 1).Value = row - 5

        # ... update the uptime of the machine in the Excel table
        up_time = str(objects.get_child([idx+":Machine", idx+":UpTime"]).get_value())
        excel_sheet.Cells(row, 2).Value = up_time
        excel_sheet.Cells(2, 4).Value = up_time

        # ... update the number of produced products in the Excel table
        product_count = str(objects.get_child([idx + ":Machine", idx + ":ProductCounter"]).get_value())
        excel_sheet.Cells(row, 3).Value = product_count
        excel_sheet.Cells(3, 4).Value = product_count

        # for the next entry: update the row counter
        row += 1

finally:
    # in any case disconnect from server and close Excel
    proxy.disconnect()
    excel_file.Close(False)
    excel_application.Application.Quit()
```

pip install --user pywin32

Agenda

Ausblick

- Zugriff auf Siemens-Steuerungen (S7)

Ubuntu

If you are using Ubuntu you can use the Ubuntu packages from our launchpad PPA. To install:

```
$ sudo add-apt-repository ppa:gijzelaar/snap7
$ sudo apt-get update
$ sudo apt-get install libsnap7-1 libsnap7-dev
```

- MODBUS TCP …
- MQTT …

- **Security: Nutzen Sie FreeOPCUA im Intranet oder verschlüsseln Sie die Verbindung!**
  - https://github.com/FreeOpcUa/python-opcua/blob/master/examples/server-with-encryption.py
  - https://github.com/FreeOpcUa/python-opcua/blob/master/examples/client-with-encryption.py

Kontakt

Mario Thron (Referent)
mario.thron@ifak.eu

Dr. Matthias Riedl
matthias.riedl@ifak.eu

Mittelstand Digital Zentrum Magdeburg
info@vernetzt-wachsen.de
www.digitalzentrum-magdeburg.de

ifak

ifak Institut für Automation und Kommunikation e.V.
Werner-Heisenberg-Str. 1
39106 Magdeburg
Tel.: +49 391 990140
www.ifak.eu

Mittelstand-Digital Zentrum Magdeburg und Mittelstand-Digital

- Das **Mittelstand-Digital Zentrum Magdeburg** gehört zu **Mittelstand-Digital**. Mit Mittelstand-Digital unterstützt das Bundesministerium für Wirtschaft und Klimaschutz (BMWK) die Digitalisierung in kleinen und mittleren Unternehmen und dem Handwerk.

- Mittelstand-Digital informiert kleine und mittlere Unternehmen über die Chancen und Herausforderungen der Digitalisierung. Regionale Kompetenzzentren helfen vor Ort dem kleinen Einzelhändler genauso wie dem größeren Produktionsbetrieb mit Expertenwissen, Demonstrationszentren, Netzwerken zum Erfahrungsaustausch und praktischen Beispielen. Das Bundesministerium für Wirtschaft und Klimaschutz ermöglicht die kostenlose Nutzung aller Angebote von Mittelstand-Digital.

- Weitere Informationen finden Sie unter www.mittelstand-digital.de.