OPC UA EPICS BRIDGE

W. D. Duckitt. Stellenbosch University, Stellenbosch, South Africa

J. K. Abraham. iThemba LABS, Cape Town, South Africa

Abstract

Open Platform Communications United Architecture (OPC UA) is a service-orientated communication architecture that supports platform-independent, data exchange between embedded micro-controllers, PLCs or PCs and cloud-based infrastructure.

This makes OPC UA ideal for developing manufacturer independent communication to vendor specific PLCs, for example. With this in mind, we present an OPC UA to EPICS bridge that has been containerized with Docker to provide a micro-service for communicating between EPICS and OPC UA variables.

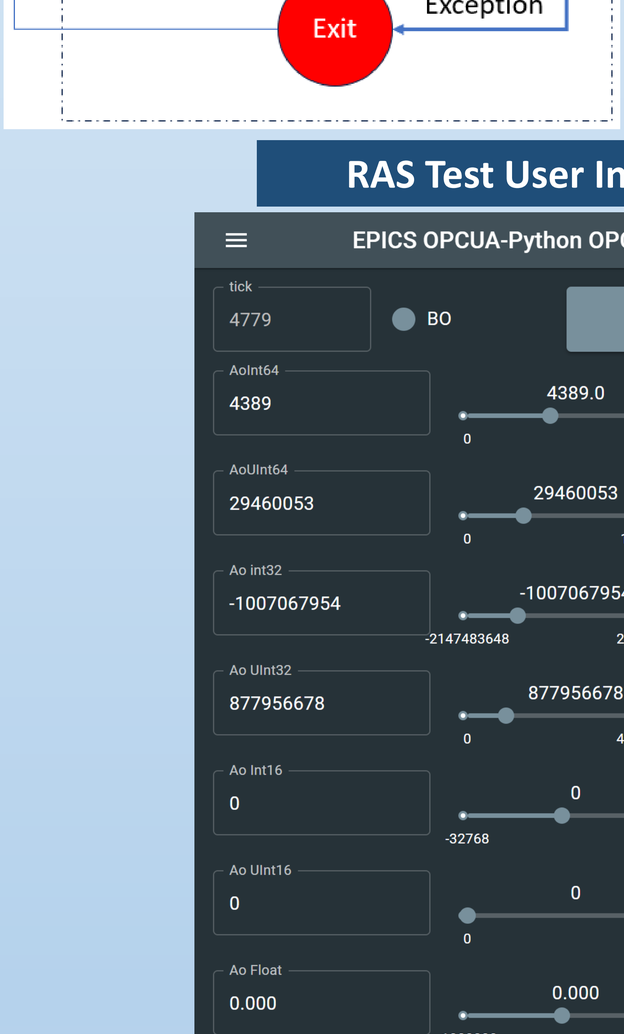
Motivation

Our experience has shown that infrastructure suppliers typically have very little knowledge of EPICS. These suppliers, also typically implement their control systems in proprietary technology from PLC manufacturers.

With all the major PLC suppliers[3–5] it is possible to install an OPC UA server within the PLC and export and serve these PLC variables over OPC UA. With our investigation, we found that it was possible to develop open-source OPC UA clients using Python[8, 9] as an alternative to the existing EPICS device support module for OPC UA[6, 7].

With this in mind, we used our experience in developing containerized EPICS systems in the React-Automation-Studio project[10] and present a system which is containerized with Docker[11], deployable as microservices and implemented in Python using the Open-Sourced Python SoftIOC project[12] and OPCUA-asyncio[9] modules.

State Diagram

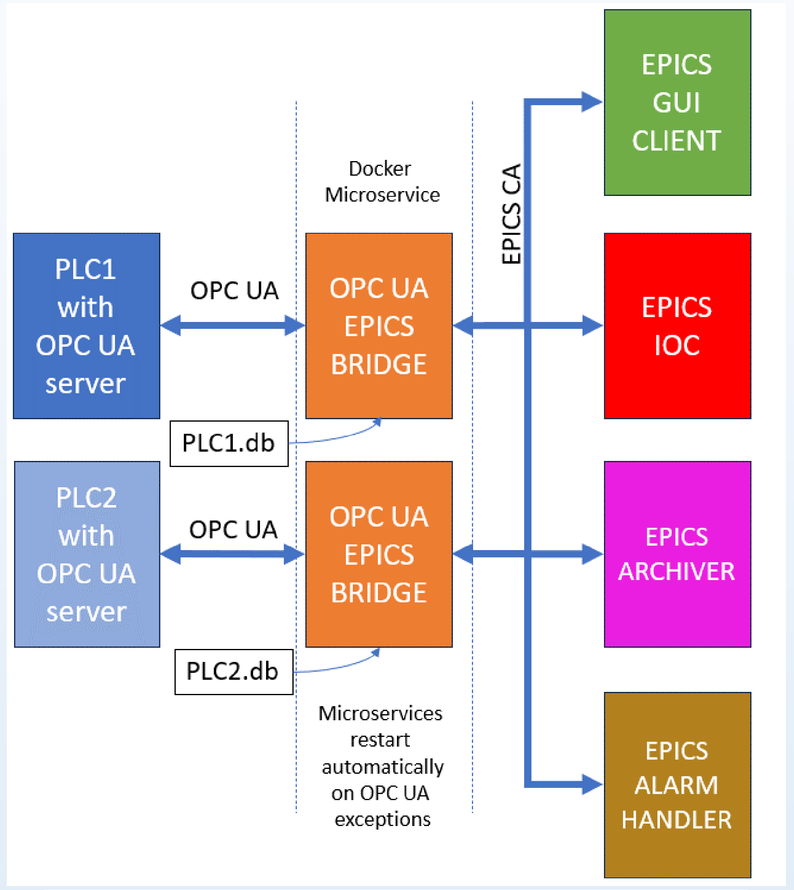


Example EPICS DB

```
record ( bo , "OPCUA−Python:Bo " )
{
        field (DTYP, " OPCUA_Boolean " )
        field (OUT, " ns =4; s=GVL.BoBool " )
        field (DESC, "BO" )
        field (ZNAM, " Off " )
        field (ONAM, "On " )
}
record ( ao , "OPCUA−Python:AoUInt16 " )
{
        field (DTYP, " OPCUA_UInt16 " )
        field (OUT, " ns =4; s=GVL. AoUInt16 " )
        field (DESC, "Ao UInt16 " )
        field (HOPR, " 65535 " )
        field (LOPR, " 0 " )
        field (PREC, " 0 " )
}
```

RAS Test User Interface

Implementation Diagram



Example Docker Compose Config

```
version: '3.2'
services:
  opcuapepicsbridge:
    build:
      context: ./
      dockerfile: docker/opcuaEpicsBridge/Dockerfile
    restart: always
    network_mode: host
    tty: true
    stdin_open: true
    environment:
      - name=OpcuaTest1
      - url=opc.tcp://192.168.56.104:4840
      - subscriptionRate=100
      - secure=False
    volumes:
      - ./certificates:/certificates/
      - ./db/testBeckhoff.db:/bridge/bridge.db
```

OPC UA, PLC and EPICS Compatibility

OPCUA Data Type	PLC Data Type	New OPCUA Bridge EPICS DTYP	Compatible EPICS Records					
			AI	AO	BI	BO	STRINGIN	STRINGOUT
Boolean	BOOL	OPCUA_Boolean			Y	Y		
SByte	SINT	OPCUA_SByte	Y	Y				
Byte	USINT	OPCUA_Byte	Y	Y				
Int16	INT	OPCUA_Int16	Y	Y				
Int32	DINT	OPCUA_Int32	Y	Y				
String	STRING	OPCUA_String					Y	Y
Float	REAL	OPCUA_Float	Y	Y				
Double	LREAL	OPCUA_Double	Y	Y				
UInt16	UINT	OPCUA_UInt16	Y	Y				
UInt32	UDINT	OPCUA_UInt32	Y	Y				
Int64	LINT	OPCUA_UInt64	Y	Y				
UInt64	ULINT	OPCUA_UInt64	Y	Y				
DateTime	DT	OPCUA_DateTime					Y	

Conclusion

An OPC UA to EPICS bridge that has been containerized with Docker to provide a microservice for communicating between EPICS and OPC UA variables has been designed. The system supports all the standard OPC UA data types. We urge the EPICS community to test and evaluate the system and to provide feedback via the React-Automation-Studio discussion group[19].

REFERENCES

[1] OPC UA, https://opcfoundation.org/about/opctechnologies/opc-ua/.

[2] EPICS, https://epics.anl.gov/.

[3] Siemens, SIMATIC controller - Industrial Automation Systems, https://www.siemens.com/global/en/products/automation/systems/industrial/plc.html.

[4] Beckhoff, Automation | Open, PC-based control technology| Beckhoff, https://www.beckhoff.com/en-za/products/automation/.

[5] Omron, Programmable Logic Controllers (PLC) | OMRON,https://industrial.omron.co.za/en/products/programmable-logic-controllers.

[6] R. Lange, R. Elliot, B. Kuner, K. Vestin, C. Winkler, D. Zimoch, et al., “Integrating OPC UA Devices in EPICS,” in 18th International Conference on Accelerator and Large Experimental Physics Control Systems, 2021, MOPV026.

[7] R. Lange, epics-modules/opcua: EPICS Device Support for OPC UA, https://github.com/epics-modules/opcua.

[8] Python, https://www.python.org/.

[9] Opcua-Asyncio, https://github.com/FreeOpcUa/opcuaasyncio.

[10] W. Duckitt and J. Abraham, “React Automation Studio: A New Face to Control Large Scientific Equipment,” in Proc. Cyclotrons’19, Cape Town, South Africa, 2020, pp. 285–288. doi:10.18429/JACoW-Cyclotrons2019-THA03

[11] Docker, https://www.docker.com/.

[12] pythonSoftIOC, https://dls-controls.github.io/pythonSoftIOC/master/index.html.

[13] Git, https://git-scm.com/.

[14] dbtoolspy: Python Module to Read EPICS Database, https://github.com/paulscherrerinstitute/dbtoolspy.

[15] Docker Compose, https://docs.docker.com/compose/

[16] Kubernetes, https://kubernetes.io/

[17] J. A. W. Duckitt, React-Automation-Studio/OPCUA-EPICSBRIDGE, https://github.com/React-Automation-Studio/OPCUA-EPICS-BRIDGE.

[18] TwinCAT 3 OPC UA TF6100, https://www.beckhoff.com/enza/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-connectivity/tf6100.html.

[19] React-Automation-Studio · Discussions · GitHub, https://github.com/orgs/React-Automation-Studio/discussions.