LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

OPC
FOUNDATION

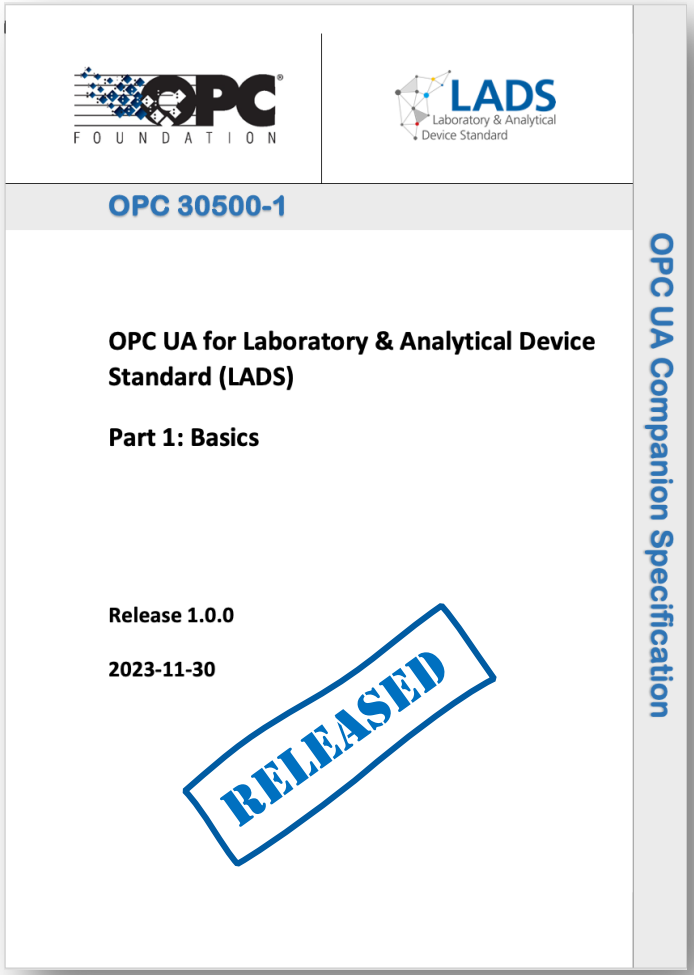
LADS OPC UA
Modeling & Implementation Workshop

From device data sheet to LADS OPC UA server

Dr. Matthias Arnold

December 2023

LADS
Laboratory & Analytical
Device Standard



ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

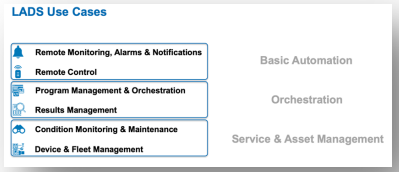
OPC FOUNDATION

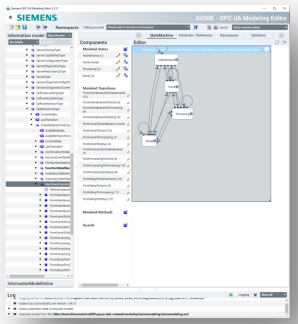
VDMA

LADS
Laboratory & Analytical Device Standard

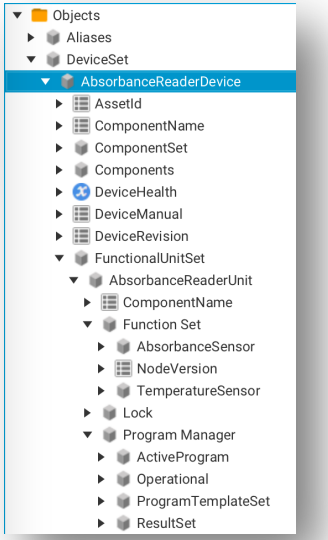
ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

From Device to LADS OPC UA server





```
const nodeset_lads = path.join(nodeset_path, 'lads.xml')
const nodeset_platereadertest = path.join(nodeset_path, 'absorbancereader.xml')
try {
    const server = new OPCUAServer({
        port: 26543, buildInfo: {
            manufacturerName: "SPECTARIS", productName: "Platereader test server",
        },
        nodeset_filename: [
            nodesets.standard,
            nodesets.di,
            nodesets.machinery,
            nodeset_lads,
            nodeset_platereadertest
        ]
    });
    await server.start();
```



Device → Information & Capabilities → Modeling → LADS OPC UA Device Model → LADS OPC UA Server

LADS OPC UA node-set

LADS device node-set

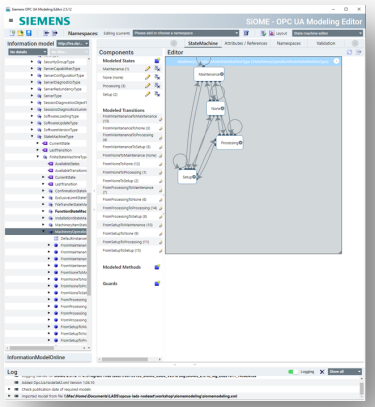
OPC UA Software Stack

LADS
Laboratory & Analytical Device Standard

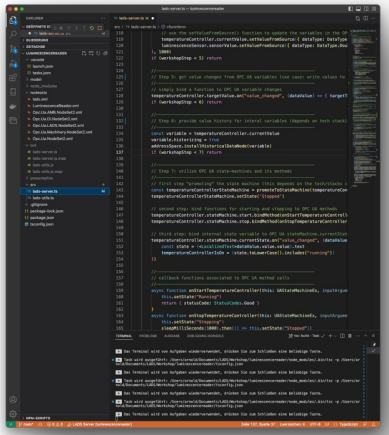
ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Today's Tools & Environment

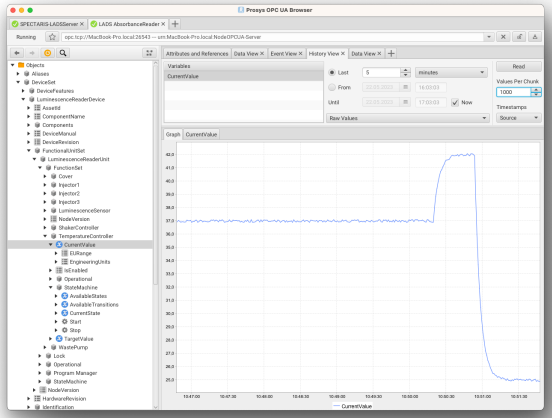
Siemens SiOME Modeler



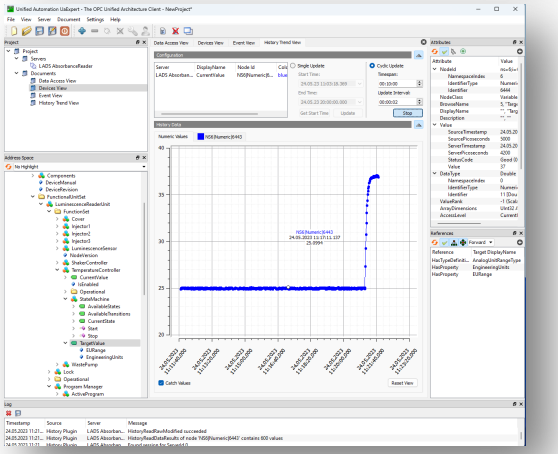
Microsoft Visual Studio Code



prosys OPC UA Browser



Unified Automation UAExpert



OPC FOUNDATION

LADS & OPC UA node-sets
GitHub OPC Foundation

node JS

TS

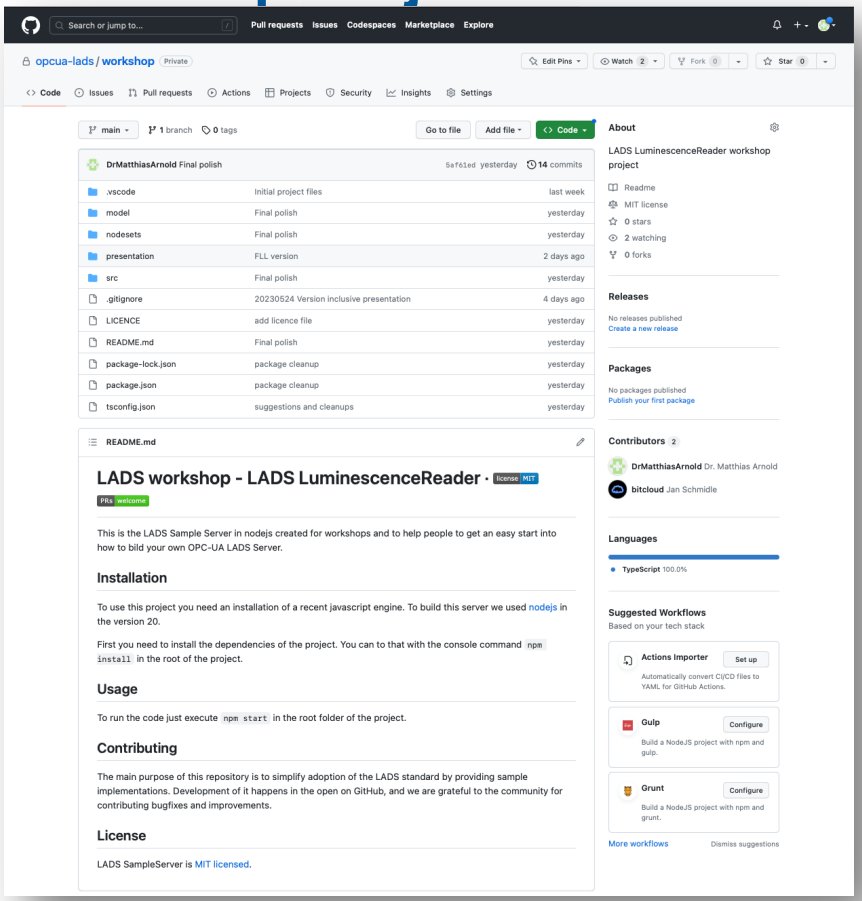
node JS
open source
for industrial
interoperability
OPC UA

Node.js, Typescript,
node-opcua
npm install _

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS OPC UA Workshop Project available on GitHub



https://github.com/opcua-lads/workshop

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

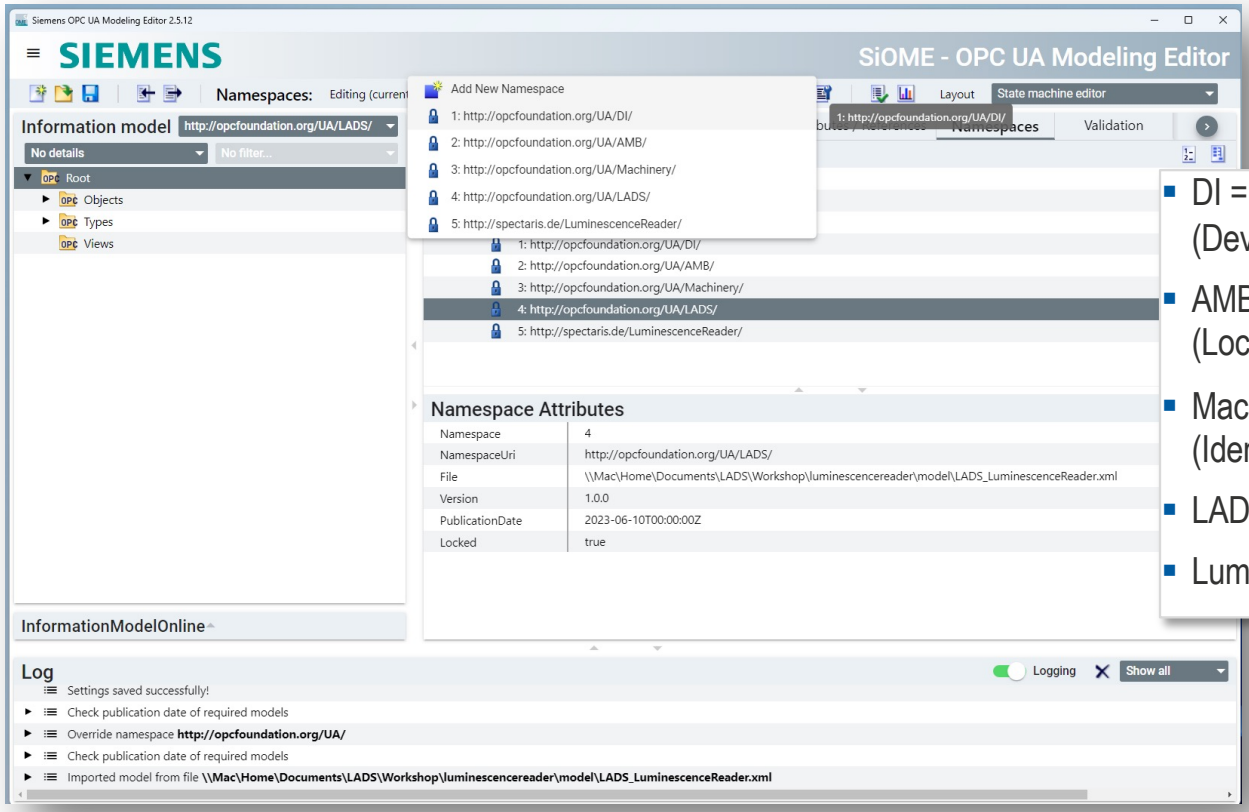
Step 0: Modeling your device with LADS OPC UA

From device data-sheet to LADS OPC UA information model …

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Preparing the Modeler and the OPC UA “Libraries”

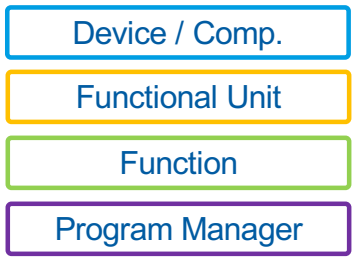


- DI = Device Integration
  (Device, Component, Nameplate, Condition Monitoring, ..)
- AMB = Asset Management Basics
  (Locations, Maintenance Tasks, ..)
- Machinery = Machinery Basics
  (Identification, ..)
- LADS = Laboratory & Analytical Device Standard
- LuminescenceReader = “Your Device”

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS Devices, Components, Functional Units, Functions, ..



A **Device** is tangible and composed of tangible **Components**.

A **Functional Unit** aggregates functions to achieve a specific outcome and is typically utilized by only one user at a time.

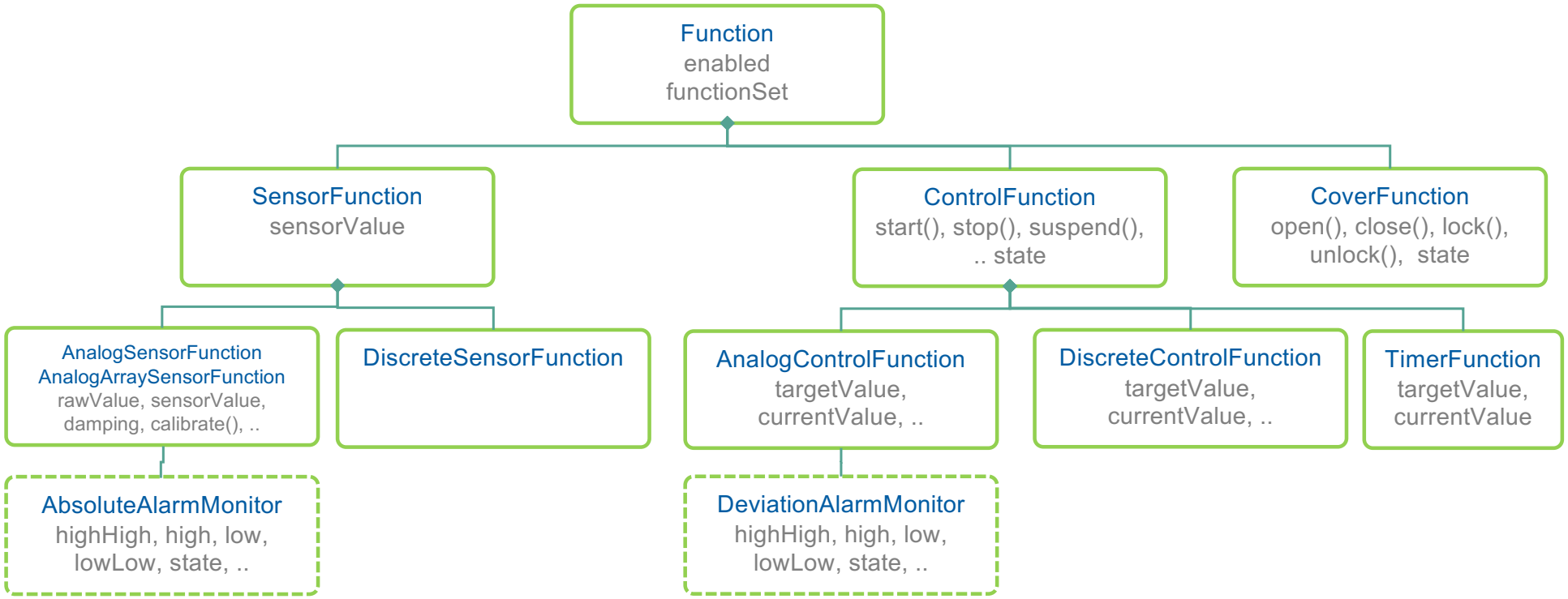
Functions are organized by a Functional Unit and used to achieve specific outcomes (“do the job”). They might utilize one ore more tangible components.

A **Program Manager** organizes the objects required to manage program templates, run programs and manage their results

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY in the German Industry Association SPECTARIS

LADS Functions (excerpt)



LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Model your device with LADS OPC UA patterns..

Centro LB 963 Microplate Luminometer

The Centro is a high-performance, easy to use microplate luminometer for both flash, and glow luminescence applications.

The optimized design provides excellent performance and flexibility:

- Superior sensitivity (<1.8 zmol firefly luciferase)
- Negligible crosstalk (10^{-6})
- Built-in shaker
- JET-injectors
- Temperature control (model-dependent)
- Ergonomic design
- Automation compatibility

The Centro is ideally suited for all luminescent reporter gene assays, immunoassays (LIA, ILMA), cell-based, and biochemical assays.

To meet your compliance requirements, a set of validation tools and optional software providing 21 CFR part 11 compliance are available.

BERTHOLD

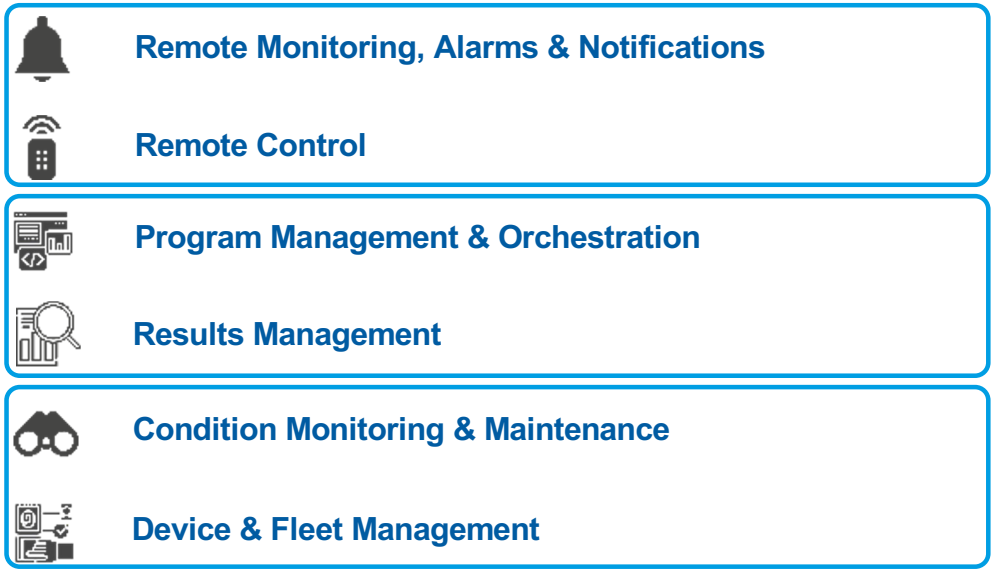
Technical Specifications

Detection unit	Low-noise photomultiplier tube in single photon counting mode Spectral range: 340 - 630 nm
Sensitivity	< 1.8 amol ATP (Centro XS) < 1.8 zmol firefly lucif. (Centro XS) < 5 amol ATP (Centro) < 2 zmol firefly luciferase (Centro)
Dynamic range	> 6 orders of magnitude
Crosstalk	Low crosstalk due to crosstalk reduction design: 10^{-6}
Injection Unit	Up to 3 injectors, JET-injection technology Variable volumes: 10 - 100 µL Speed 200 - 440 µL/sec Accuracy better 2% (over entire range of volume) Precision better 2% (over entire range of volume)
Shaking	3 modes, variable amplitude and speed
Temperature Control	+5°C above room temperature up to 45°C
Microplate Formats	96-well, solid and strip, dimensions 128.2 x 86.0 x 14.7 mm (L x W x H)

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS Use Cases



Basic Automation

Orchestration

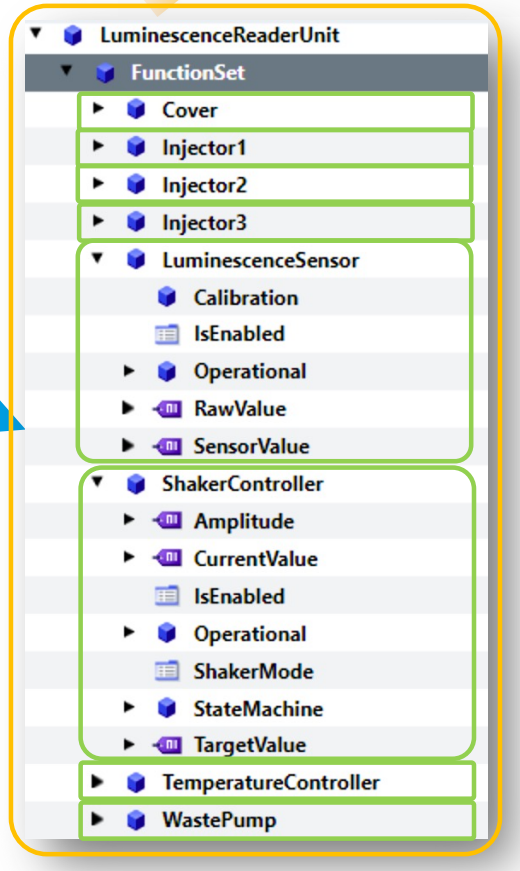
Service & Asset Management

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS Use Cases & Modeling

- Remote Monitoring, Alarms & Notifications
- Remote Control
- Program Management & Orchestration
- Results Management
- Condition Monitoring & Maintenance
- Device & Fleet Management

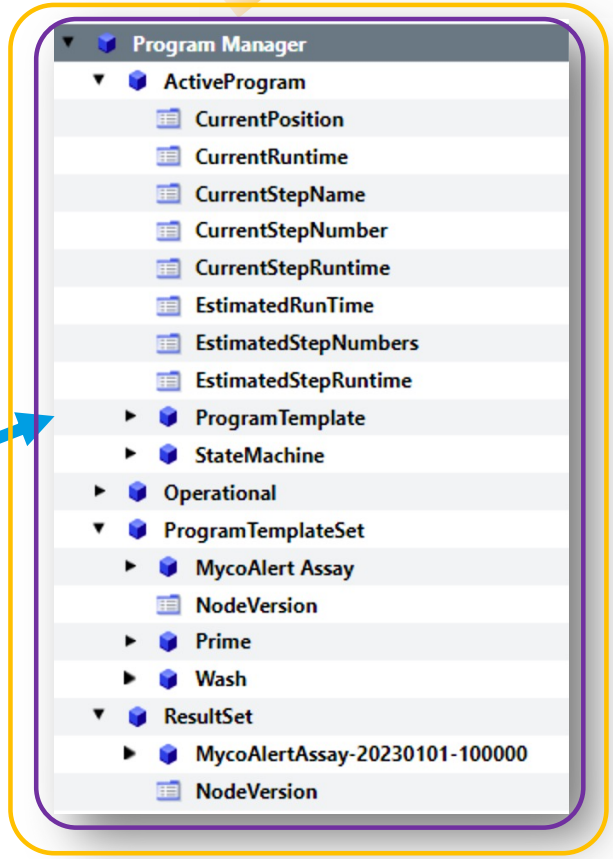


LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS Use Cases & Modeling

- Remote Monitoring, Alarms & Notifications
- Remote Control
- Program Management & Orchestration
- Results Management
- Condition Monitoring & Maintenance
- Device & Fleet Management

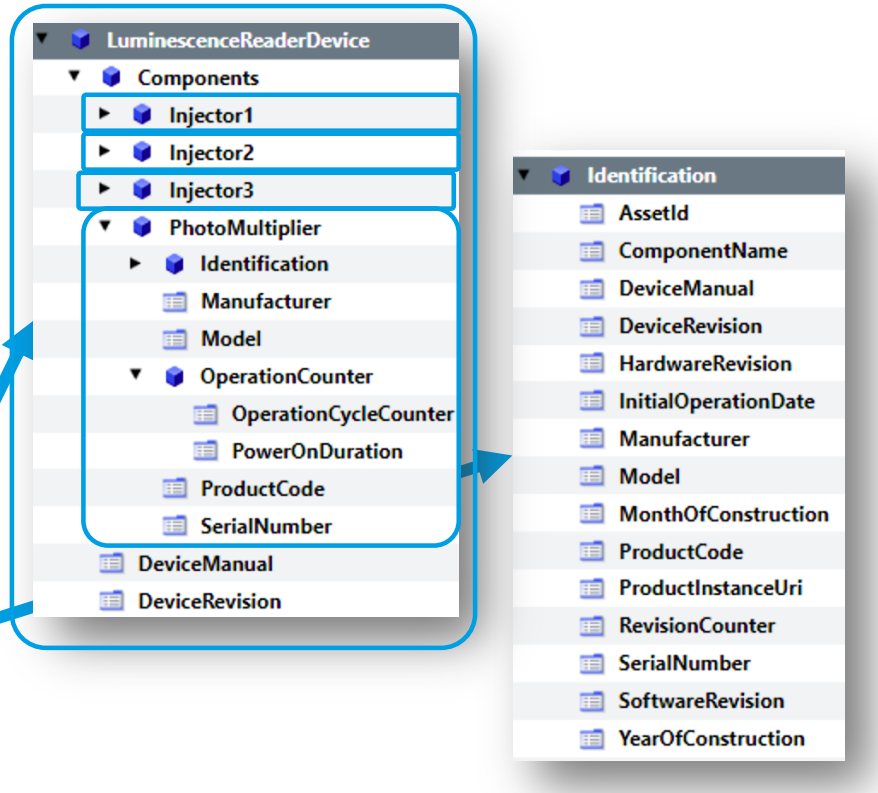


LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

LADS Use Cases & Modeling

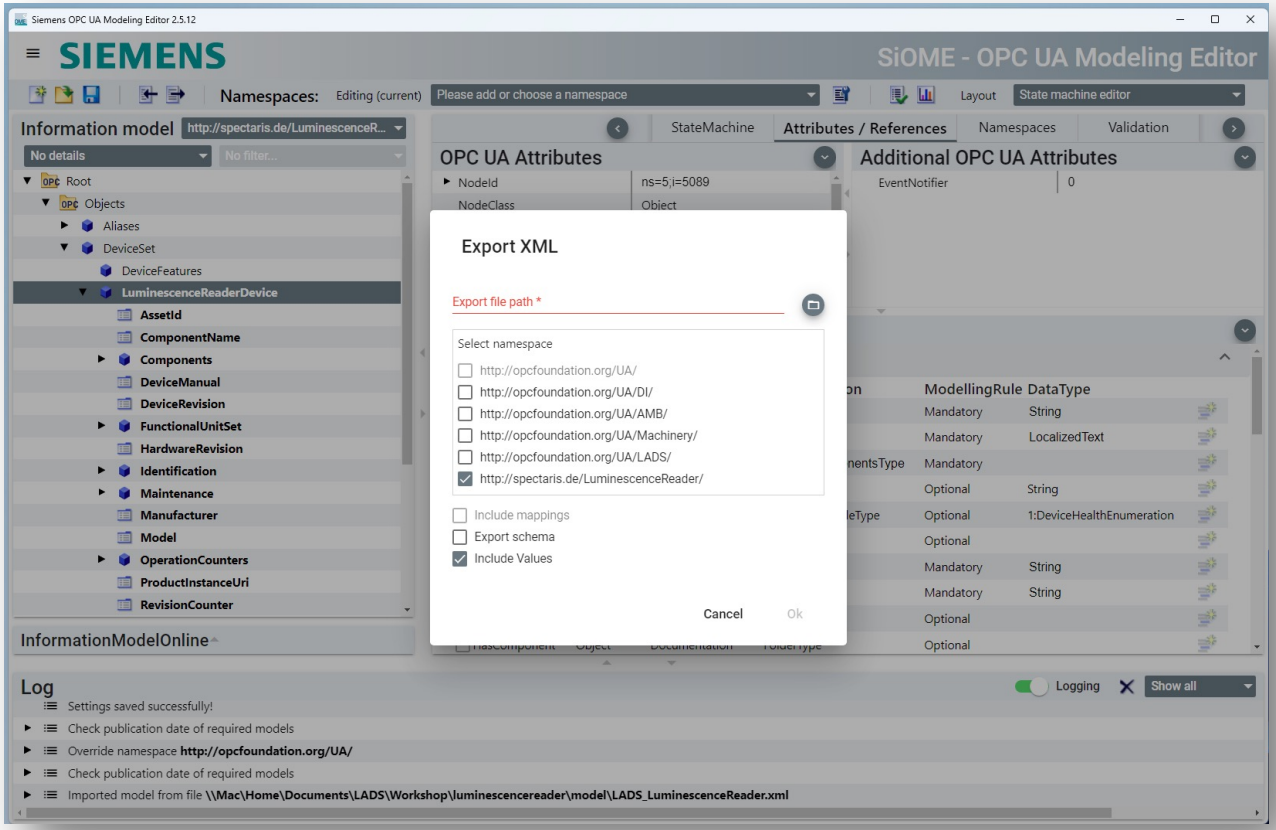
- Remote Monitoring, Alarms & Notifications
- Remote Control
- Program Management & Orchestration
- Results Management
- Condition Monitoring & Maintenance
- Device & Fleet Management



LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Export your LADS OPC UA device information model as node-set



LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 1:
Importing the Information-Model into software

Towards a running LADS OPC UA server skeleton within minutes …

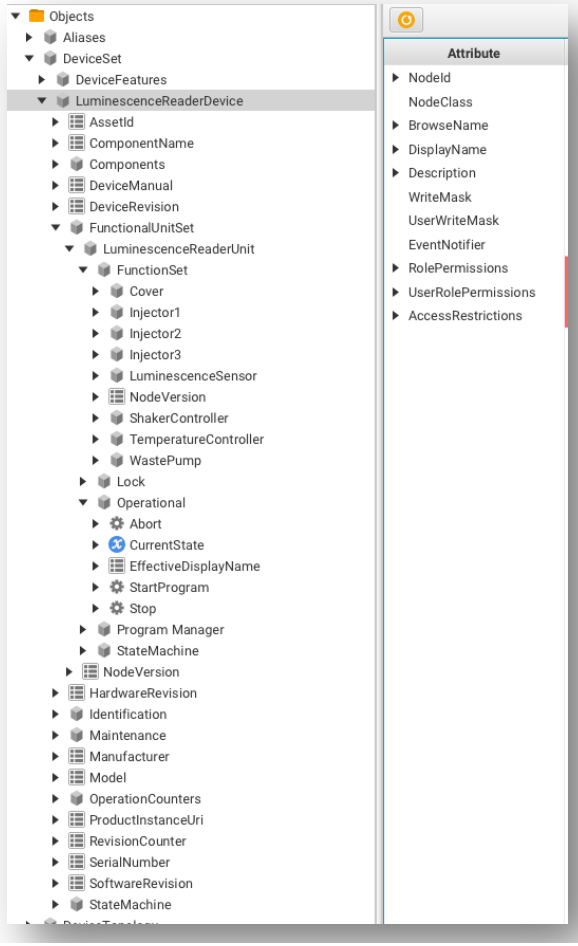
LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Import the required node-set files and start the OPC UA server

```
//------------------------------------------------------------------
// Step 1: load the required OPC UA nodesets and start the server
//------------------------------------------------------------------
// provide paths for the nodeset files
//const nodeset_path = './src/workshop/luminescencereader/nodesets'
const nodeset_path = join(__dirname, '../nodesets')
const nodeset_standard = join(nodeset_path, 'Opc.Ua.NodeSet2.xml')
const nodeset_di = join(nodeset_path, 'Opc.Ua.DI.NodeSet2.xml')
const nodeset_amb = join(nodeset_path, 'Opc.Ua.AMB.NodeSet2.xml')
const nodeset_machinery = join(nodeset_path, 'Opc.Ua.Machinery.NodeSet2.xml')
const nodeset_lads = join(nodeset_path, 'Opc.Ua.LADS.NodeSet2.xml')
const nodeset_luminescencereader = join(nodeset_path, 'LuminescenceReader.xml')
try {
    // build the server object
    const server = new OPCUAServer({
        port: 26543, buildInfo: {
            manufacturerName: "SPECTARIS",
            productUri: "",
            softwareVersion: "1.0.0",
        },
        serverInfo: {
            applicationName: "LADS LuminescenceReader",
        },
        nodeset_filename: [
            nodeset_standard,
            nodeset_di,
            nodeset_machinery,
            nodeset_amb,
            nodeset_lads,
            nodeset_luminescencereader,
        ]
    })

    // start the server
    await server.start();
    const endpoint = server.endpoints[0].endpointDescriptions()[0].endpointUrl; console.log(" server is ready on "
    console.log("CTRL+C to stop");
```



LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 2:
Find and list devices from the OPC UA DeviceSet

Where to find devices within my code ?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Get the OPC UA DeviceSet and browse its children (= Devices)

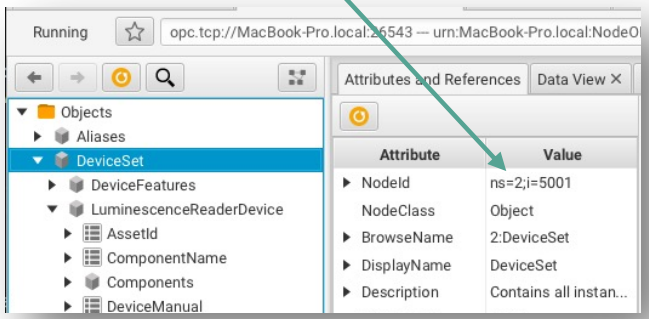
```
//-----------------------------------------------------------------
// Step 2: search for devices availabe in the server's DeviceSet
// DeviceSet is defined by OPC UA Device Integration and represents the collection of devices in a server
//-----------------------------------------------------------------
const addressSpace = server.engine.addressSpace
const nameSpaceDI = addressSpace.getNamespace('http://opcfoundation.org/UA/DI/')
const deviceSet = <UAObject>nameSpaceDI.findNode(coerceNodeId(5001, nameSpaceDI.index))
const deviceReferences = deviceSet?.findReferencesExAsObject(coerceNodeId(ReferenceTypeIds.Aggregates, 0))
const devices = deviceReferences?.map((device) => {return <UADevice>device})

devices.forEach((device: UADevice) => {
    console.log(`Found device ${device.browseName} of type ${device.typeDefinitionObj.browseName}`)
})
```

```
/usr/local/bin/node ./out/lads-server.js
 server is ready on  opc.tcp://MacBook-Pro.local:26543
CTRL+C to stop
Found device 6:LuminescenceReaderDevice of type 6:LuminescenceReaderDeviceType
```

1a. get the address-space and the DI namespace

1b. find the well-known DeviceSet



2a. find children of DeviceSet

2b. iterate devices and print on Console

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 3:
Build a “convenience” interface & access your device

How can a I access “my” device object?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Build “your” device interface and assign it to “your” device

```
//---------------------------------------------------------------
// Step 3: access the device as LuminescenseReader
//---------------------------------------------------------------
// first step: define an interface to conveniently access and bind the OPC UA objects
interface LuminescenceReaderFunctionalUnit extends LADSFunctionalUnit {
    functionSet: {
        luminescenceSensor: LADSAnalogArraySensorFunction
        temperatureController: LADSAnalogControlFunction
        injector1: LADSAnalogControlFunction
        injector2: LADSAnalogControlFunction
        injector3: LADSAnalogControlFunction
        cover: LADSCoverFunction
    }
}
interface LuminescenceReaderDevice extends LADSDevice {
    functionalUnitSet: {
        luminescenceReaderUnit: LuminescenceReaderFunctionalUnit
    }
}
// second setp: find device with matching type and cast to interface
const luminescenceReaderDevice = <LuminescenceReaderDevice>devices.find(
    (device) => (device.typeDefinitionObj.browseName.name.includes('Luminescence')))

// alternative pragmatic way completley omitting step 2: if the node-id is known directly acess device object
const nameSpaceLR = addressSpace.getNamespace('http://spectaris.de/LuminescenceReader/')
const _luminescenceReaderDevice = <LuminescenceReaderDevice>addressSpace.findNode(coerceNodeId(5089, nameSpaceLR.index))
```

1a. Define the FunctionSet of “your” FunctionalUnit using LADS Functions

1b. Put ”your” FunctionalUnit in “your” Device

2. Find a matching device and assign it to “your” LuminescenceReaderDevice

3. Or - skip reading DeviceSet and directly access “your” device via its node-id ;-)

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 4:
Reading device variable live-data via OPC UA

How can I send my internal variable values to OPC UA variables?

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY in the German Industry Association SPECTARIS

Pushing “your” variable values to the world..

```
//-------------------------------------------------------------------
// Step 4: set OPC UA variable values from internal variables (use case: read values from the device)
//-------------------------------------------------------------------
// first step: get selected LADS OPC UA objects using interface definition
const functionalUnit = luminescenceReaderDevice.functionalUnitSet.luminescenceReaderUnit
const functionSet = functionalUnit.functionSet
// luminescence sensor
const luminescenceSensor = functionSet.luminescenceSensor
const wells = 96
// temperature controller
const temperatureController = functionSet.temperatureController
let targetTemperature = 37.0
let currentTemperature = 25.0
let temperatureControllerIsOn = true
const damping = 0.8

// second step: periodically calulate values and update their OPC UA peer variables
setInterval(() => {
    // temperature with 1st order low pass filter and noise
    const temperature = temperatureControllerIsOn ? targetTemperature : 25
    currentTemperature = damping * currentTemperature + (1 - damping) * temperature
    // array of luminescence readings with some noise
    const luminescence = new Float64Array(wells).map((_, index) => { return index * index + (Math.random() - 0.5) })

    // use the setValueFromSource() function to update the variables in the OPC UA information model
    temperatureController.currentValue.setValueFromSource({ dataType: DataType.Double, value: currentTemperature + 0.2 * (Ma
    luminescenceSensor.sensorValue.setValueFromSource({ dataType: DataType.Double, arrayType: VariantArrayType.Array, value:
}, 1000)
if (workshopStep < 5) return
```

1a. Access the FunctionalUnit and its FunctionSet

1b. Access the LuminescenceSensor and TemperatureController functions

1c. Define some internal “firmware” variables

2a. Build a recurrent loop called every second

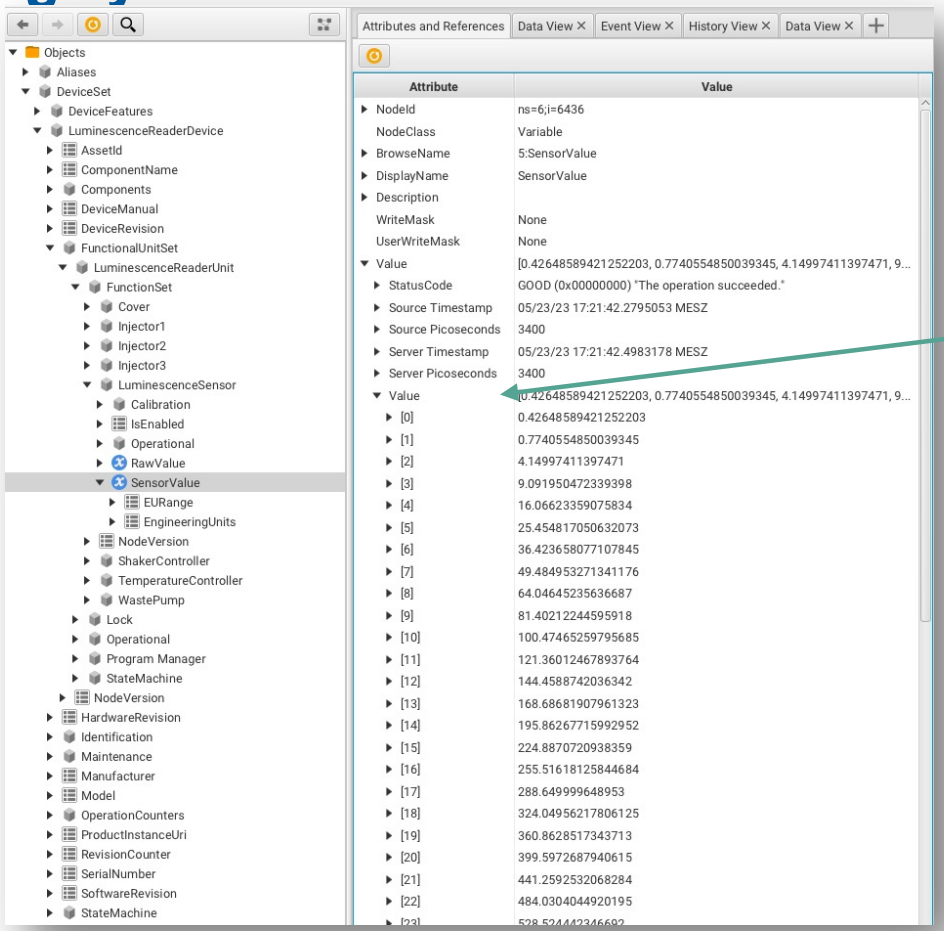
2b. Calculate simulated values for temperature and 96 luminescence readings

2b. Write “your” internal values to the corresponding OPC UA variables

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Browsing “your” variable values – Luminescence Readings

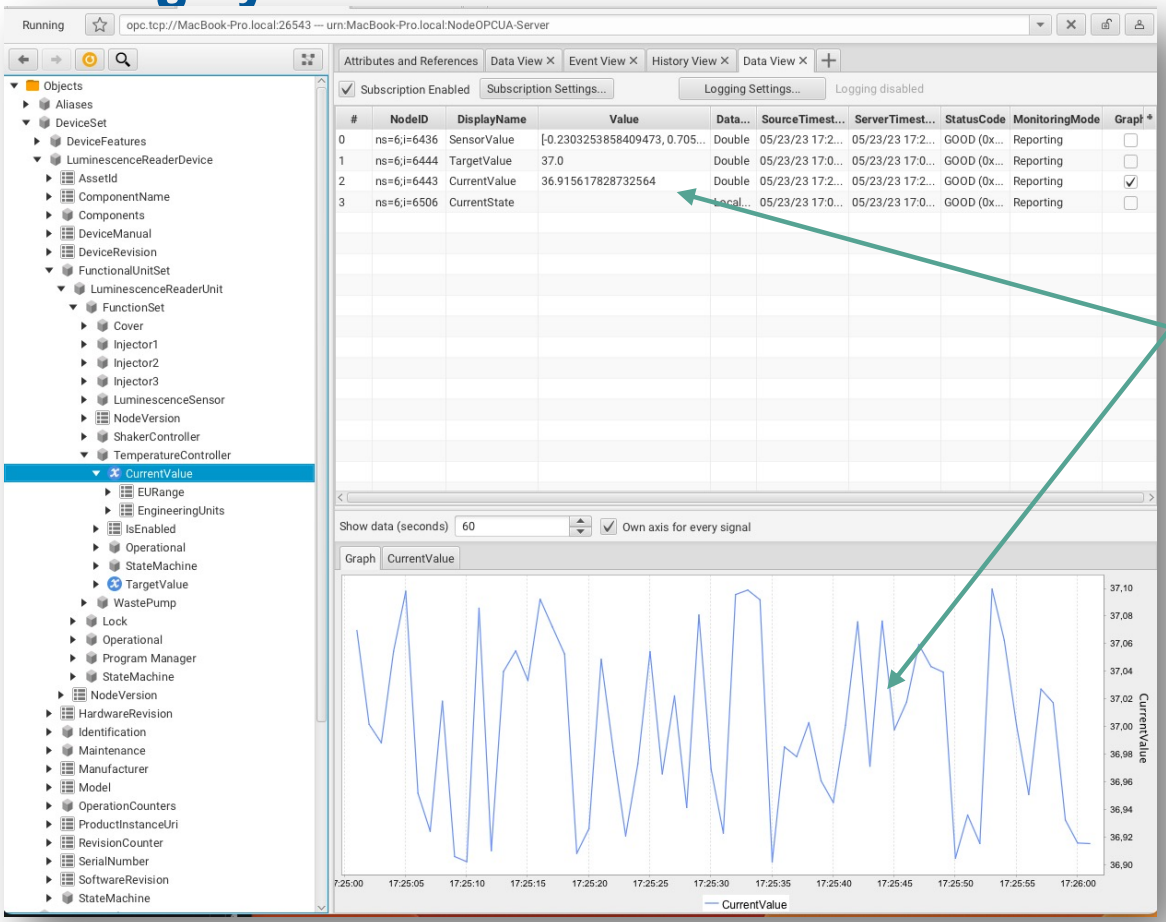


LuminescenceSensor.SensorValue
Array with 96 Luminescence Readings

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Browsing “your” variable values – Current Temperature



TemperatureController.CurrentValue
Live updates via OPC Subscription

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 5:
Writing device variable values via OPC UA

How can I write my internal variable values via OPC UA variables?

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Receiving variable values from the outside world..

```
//---------------------------------------------------------------
// Step 5: get value changes from OPC UA variables (use case: write values to the device)
//---------------------------------------------------------------
const targetValue = temperatureController.targetValue
const validateVariableValue = true
if (!validateVariableValue) {
    // simply bind a (anonymous) function to OPC UA variable changes
    targetValue.on("value_changed", (dataValue) => { targetTemperature = dataValue.value.value })
} else {
    // bind a setter / getter to the variable, do validations and return status-code
    targetValue.bindVariable({
        set: (variantValue: Variant): StatusCode => {
            const range = targetValue.euRange.readValue().value.value
            const value: number = variantValue.value
            if (range && (value > range.high) || (value < range.low)) {
                // value clamped to euRange
                const clampedValue = (value > range.high) ? range.high : range.low
                targetTemperature = clampedValue
                return StatusCodes.GoodClamped
            } else {
                // value within euRange
                targetTemperature = value
                return StatusCodes.Good
            }
        },
        get: (): Variant => { return new Variant({ dataType: DataType.Double, value: targetTemperature }) }
    })
}
```

Without validation:
Bind an (anonymous) function to OPC UA variable value changes and get the value..
That's all? That's all!

With validation:
Bind a setter/getter pair to the variable

Get the euRange property

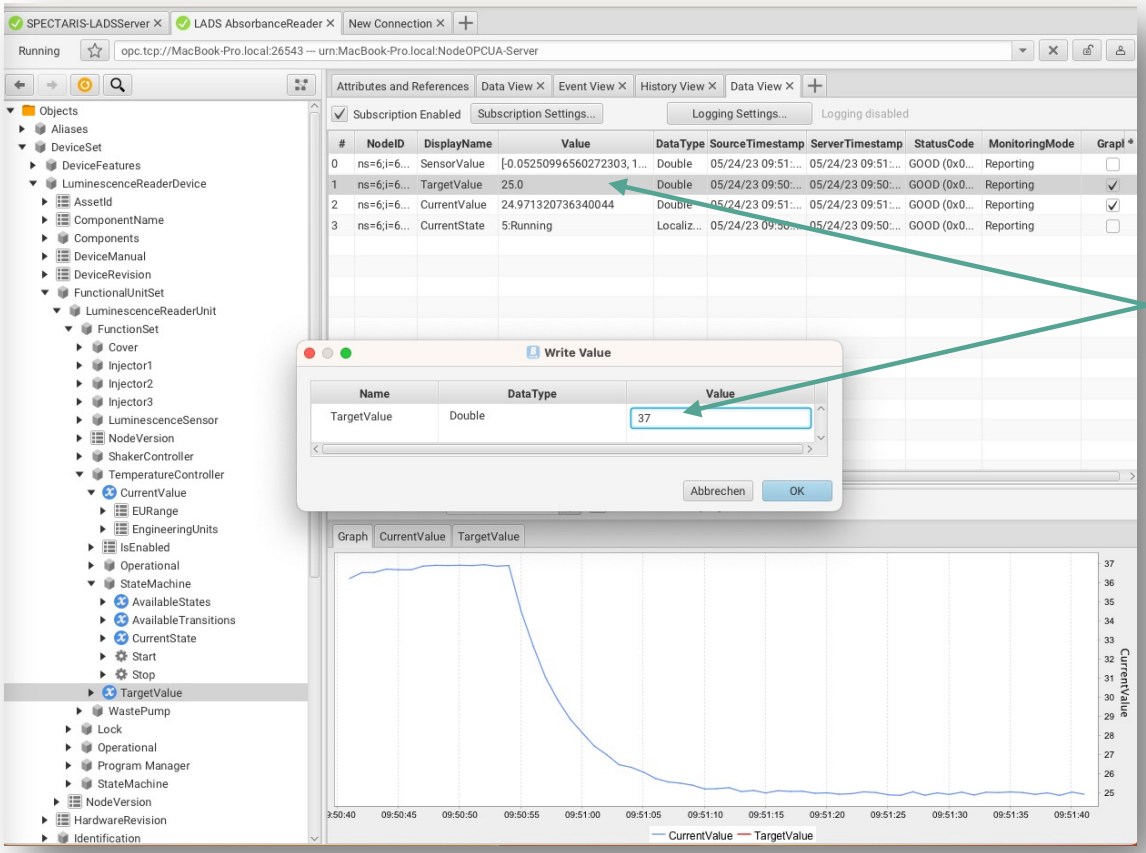
If new value exceeds range, clamp it and inform the client by returning GoodClamped

If new value within range accept it as is and return Good

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Ready to write a new temperature target-value via OPC Browser ..

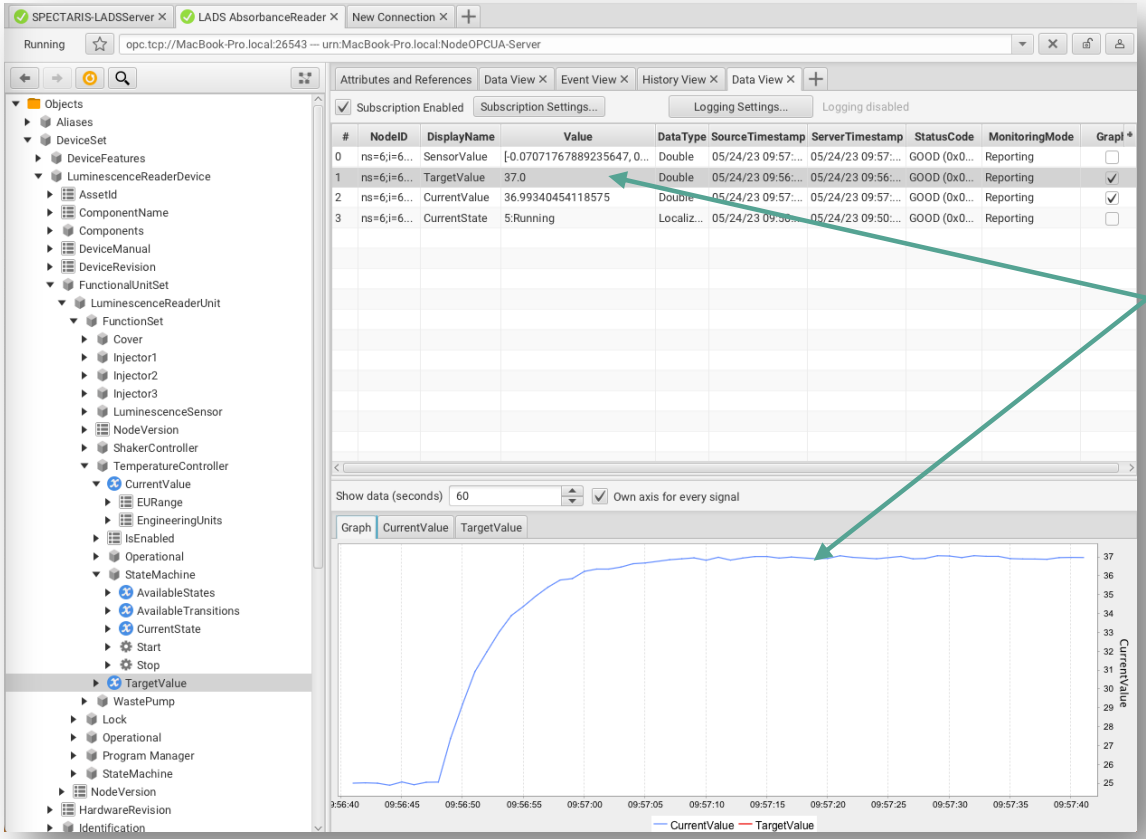


Old value: 25°C
New value: 37°C

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

New target-value consumed, and temperature is rising..



TargetValue = 37°C
CurrentValue approaching TargetValue

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 6:
Accessing variable value histories via OPC UA

How complicated is it to provide OPC UA variable value history services?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Provide variable value history services

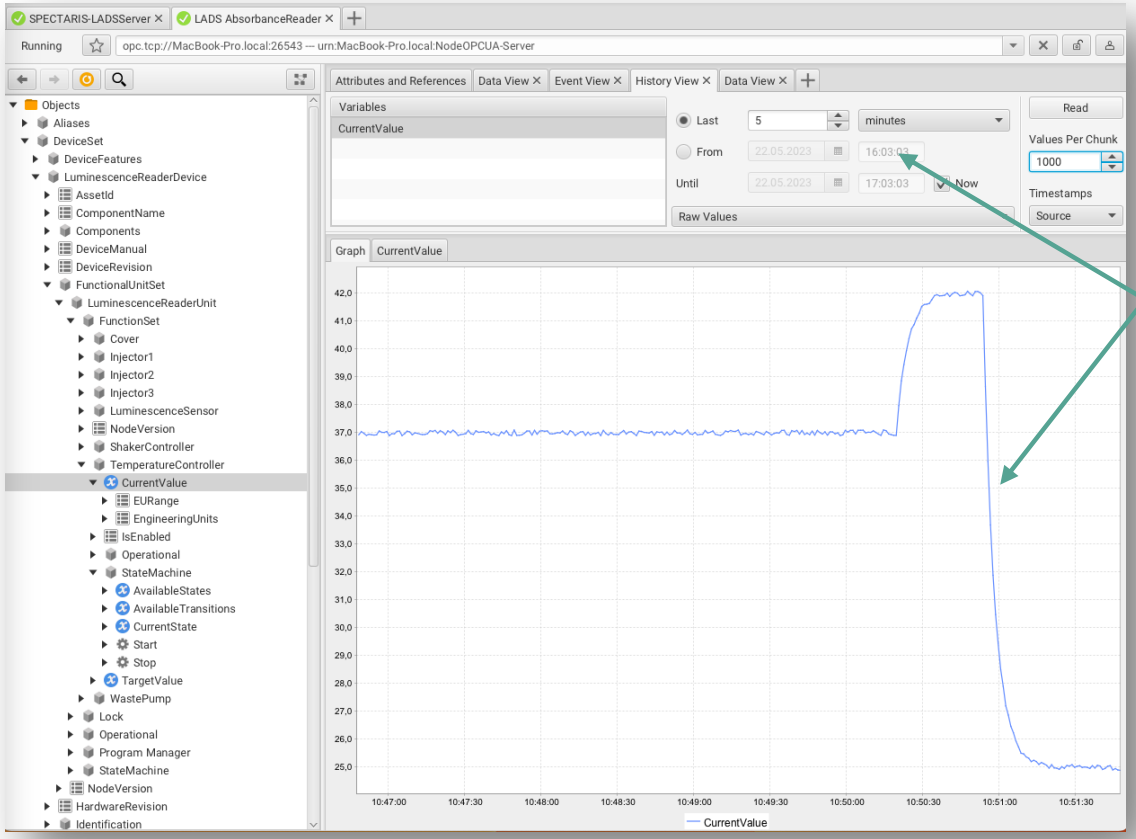
```
//-----------------------------------------------------------------
// Step 6: provide value history for interal variables (depends on tech stack)
//-----------------------------------------------------------------
const variable = temperatureController.currentValue
variable.historizing = true
addressSpace.installHistoricalDataNode(variable)
```

Mark variable as historizing (OPC UA)
and enable historian (node-opcua)
That's all? That's all!

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

History data query service – “give me the last 5 minutes ..”



Historian query interface is a standard OPC UA feature..

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

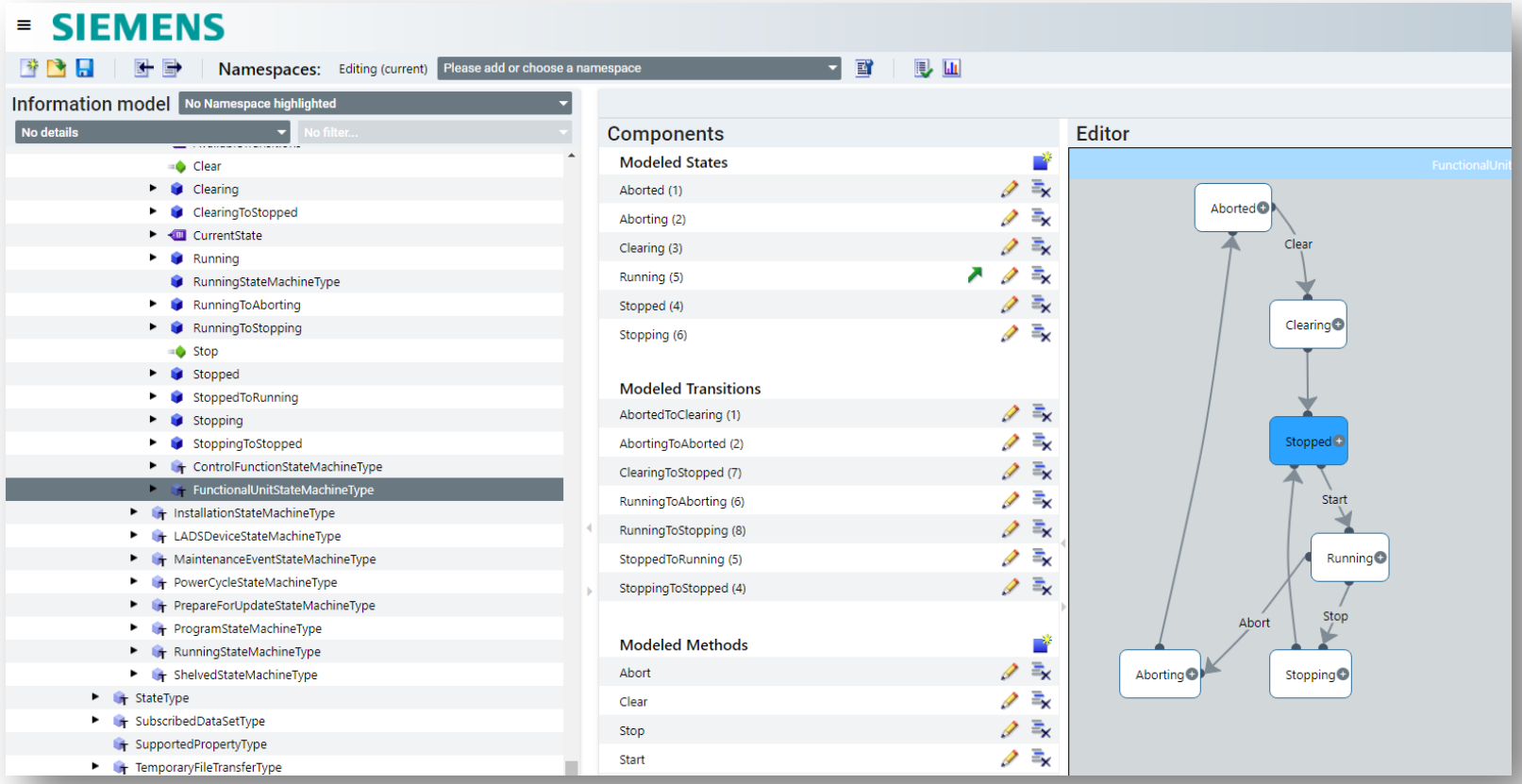
Step 7:
LADS OPC UA methods and state-machines

I want to turn on and off controllers, I want to start programs, I want to monitor their current state, ..how?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Methods & State Machines are “native” OPC UA functionality



LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Access a state-machine, bind commands and monitor state

```
//-----------------------------------------------------------------
// Step 7: utilize OPC UA state-machines and its methods
//-----------------------------------------------------------------
// first step "promoting" the state machine (this depends on the tech/stacks capabilities)
const temperatureControllerStateMachine = promoteToStateMachine(temperatureController.stateMachine)
temperatureControllerStateMachine.setState('Stopped')

// second step: bind functions for starting and stopping to OPC UA methods
temperatureController.stateMachine.start.bindMethod(onStartTemperatureController.bind(temperatureControllerStateMachine))
temperatureController.stateMachine.stop.bindMethod(onStopTemperatureController.bind(temperatureControllerStateMachine))

// third step: bind internal state variable to OPC UA stateMachine.currentState variable
temperatureController.stateMachine.currentState.on("value_changed", (dataValue: DataValue) => {
    const state = (<LocalizedText>dataValue.value.value).text
    temperatureControllerIsOn = (state.includes("Running"))
})

//-----------------------------------------------------------------
// callback functions associated to OPC UA method calls
//-----------------------------------------------------------------
async function onStartTemperatureController(this: UAStateMachineEx, inputArguments: VariantLike[], context: SessionContext): Promise<CallMethodResultOptions> {
    this.setState("Running")
    return { statusCode: StatusCodes.Good }
}
async function onStopTemperatureController(this: UAStateMachineEx, inputArguments: VariantLike[], context: SessionContext): Promise<CallMethodResultOptions> {
    this.setState("Stopping")
    sleepMilliSeconds(1000).then(() => this.setState("Stopped"))
    return { statusCode: StatusCodes.Good }
}
```

Access state-machine and “promote” it

Bind functions to method calls for start() and stop()

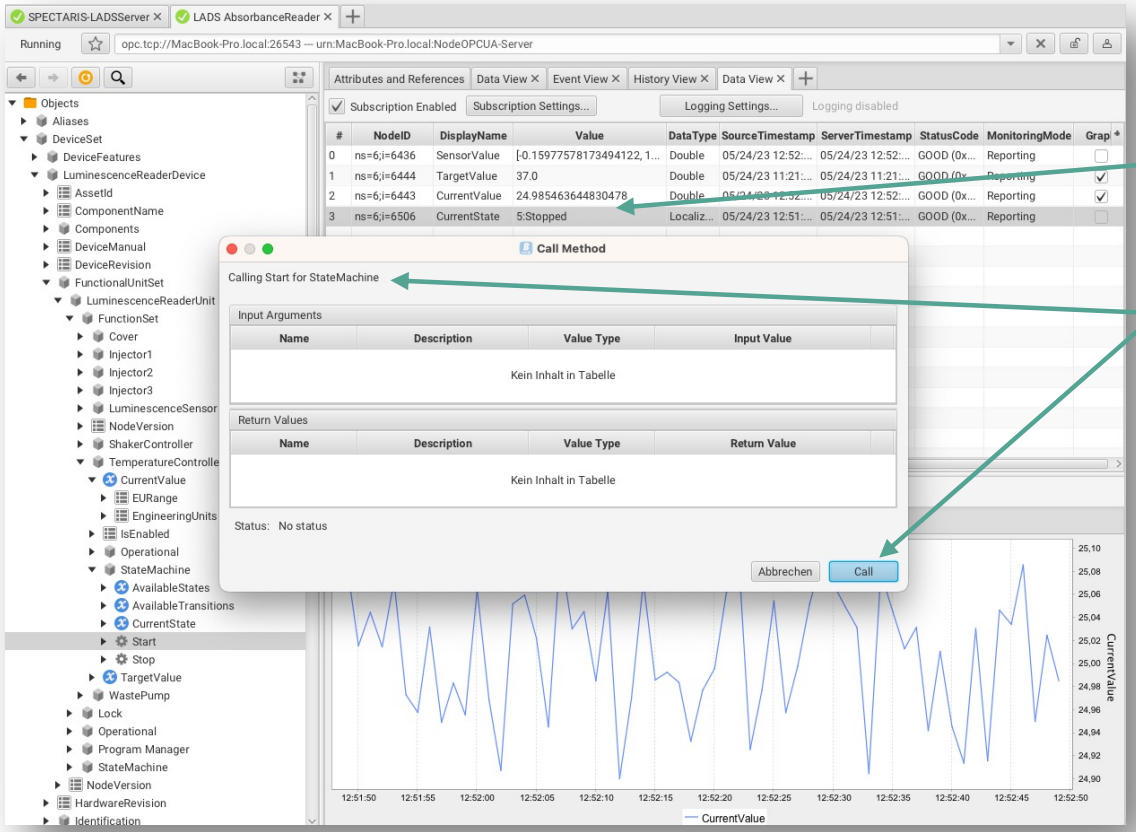
Bind internal variable to state changes

The functions which are called when methods start() and stop() are triggered..

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Ready to start the temperature-controller..



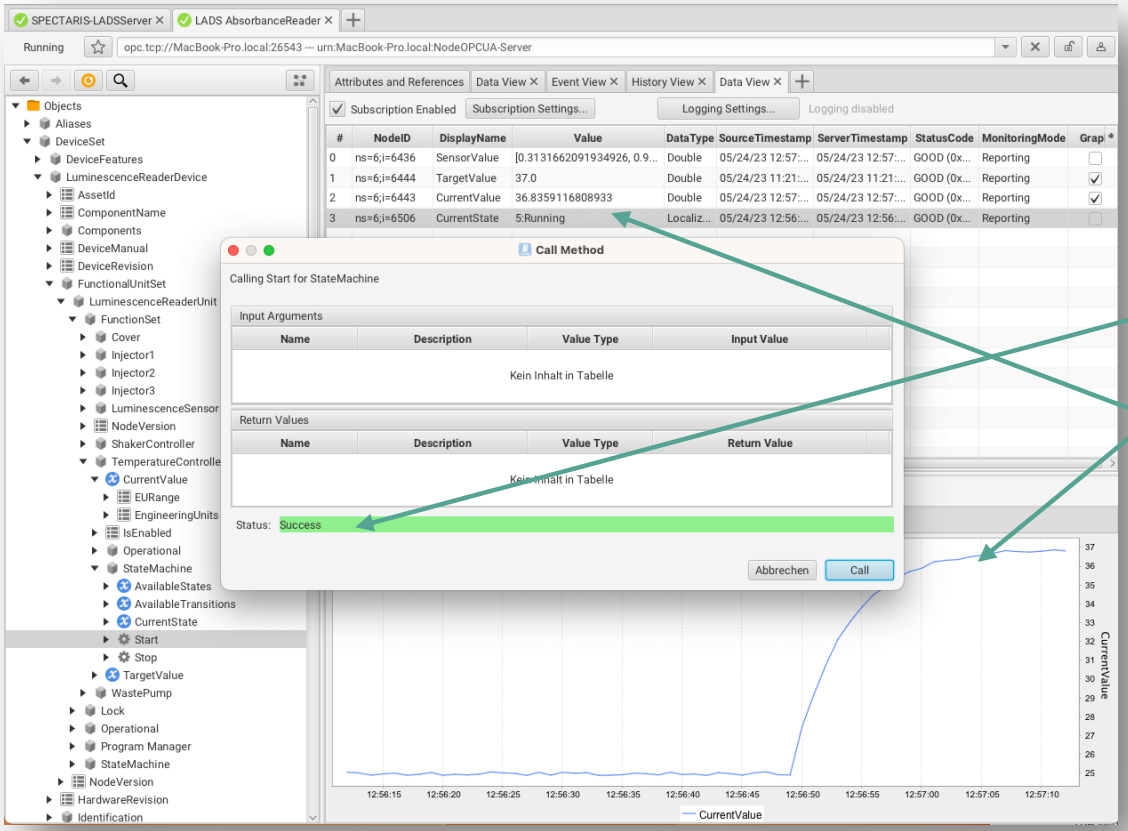
CurrentState = Stopped
CurrentValue = 25 °C

Start() method ready to be triggered..

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Temperature-controller started ..



Start() method success..

CurrentState = Running
CurrentValue = approaching 37 °C

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 8:
Event Logs & Audit Trails

Variables, histories, methods, state-machines – all nice stuff, but.
What about event-logs and audit trails?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Raising events on “whatever”-changes..

```
//---------------------------------------------------------------
// Step 8: generate OPC UA events on selected state changes (use case: Audit trails)
//---------------------------------------------------------------
// get the most basic OPC UA event type (there are many more & you can define your own event types)
const baseEventType = addressSpace.findEventType(coerceNodeId(ObjectTypeIds.BaseEventType))
// raise event whenever the state of the temperature-controller statemachine has changed
temperatureController.stateMachine.currentState.on("value_changed", (dataValue: DataValue) => {
    const message = `${temperatureController.getDisplayName()} changed state to "${(<LocalizedText>dataValue.value.value).text}".`
    temperatureController.raiseEvent(baseEventType, { message: { dataType: DataType.LocalizedText, value: message } })
})
// raise event whenever the target-value of the temperature-controller has changed
temperatureController.targetValue.on("value_changed", (dataValue: DataValue) => {
    const message = `${temperatureController.getDisplayName()} value changed to "${dataValue.value.value.toString()}".`
    temperatureController.raiseEvent(baseEventType, { message: { dataType: DataType.LocalizedText, value: message } })
})
```

Get the base-event type

Bind function to changes of current-state (just as an example)

Create a message text

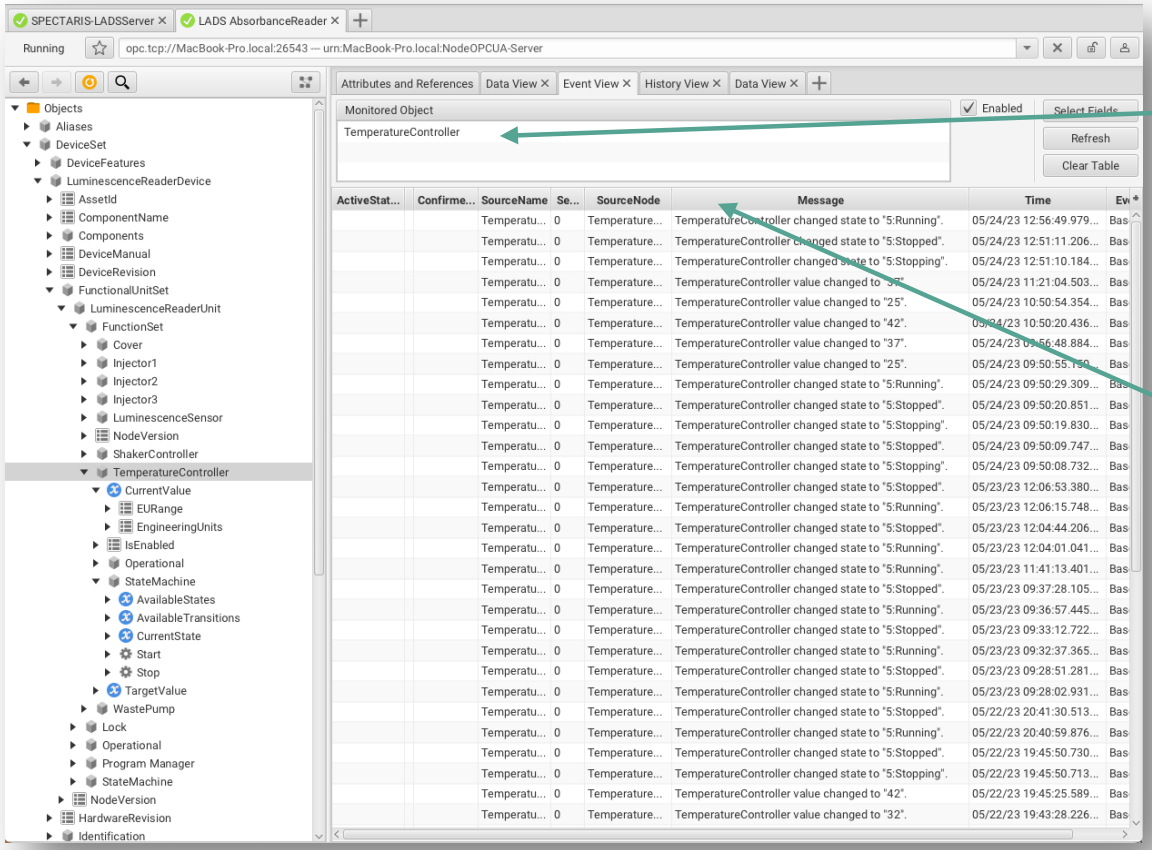
Raise the event on the source object

Raise event on change of target-value (just another example)

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Monitoring some recent temperature-controller events ..



Object to be monitored
Events might be bubbled up in the hierarchy ..
You decide ..

Base events come with timestamp, severity, message, source, ..
Complex events can have multiple fields with all kind of data..
You decide..

LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY in the German Industry Association SPECTARIS

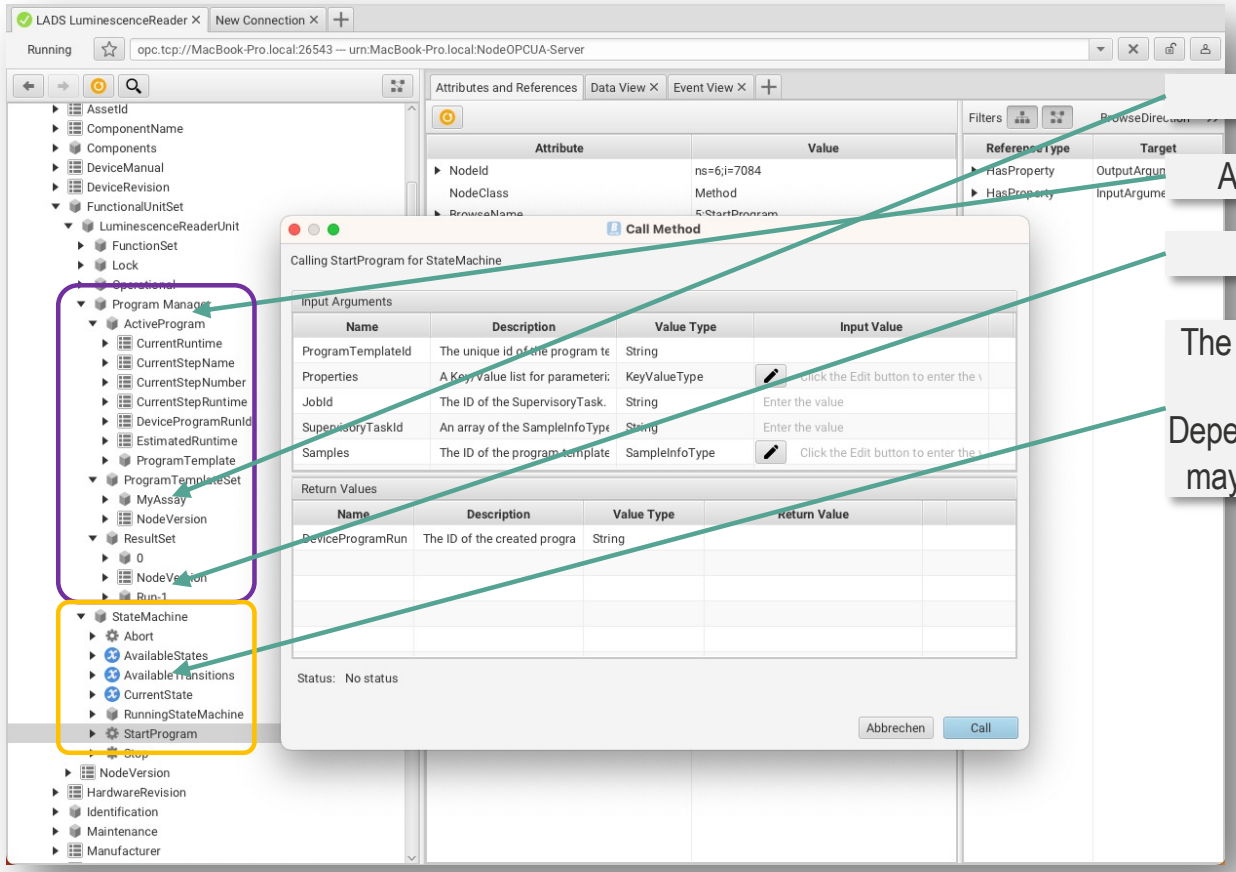
Step 9:
The big picture – running programs, storing results ..

Now that I've learned all the LADS OPC UA basics – what about orchestrating runs from remote?

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Introducing the LADS Program-Manager..



Program-templates

Active-program with runtime properties

Program-run results

The functional-unit's state-machine including state-indicator & methods.
Depending on the device type, functional-units may be operated manually or via programs.

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Starting program-runs..

```
//-------------------------------------------------------------
// Step 9: Running programs and generating results (dynamic creation of LADS OPC UA objects)
//-------------------------------------------------------------
const activeProgram = functionalUnit.programManager.activeProgram
const resultSet = functionalUnit.programManager.resultSet
const functionalUnitStateMachine = promoteToStateMachine(functionalUnit.stateMachine)
functionalUnitStateMachine.setState("Stopped")
functionalUnit.stateMachine.startProgram.bindMethod(startProgram.bind(functionalUnitStateMachine))

let runId = 0
async function startProgram(this: UAStateMachineEx, inputArguments: VariantLike[], context: SessionContext): Promise<CallMethodResultOptions> {
    // validate current state
    if (!this.getCurrentState().includes("Stopped")) { return { statusCode: StatusCodes.BadInvalidState} }

    // valdate input arguments
    inputArguments.forEach((value: VariantOptions, index) => {
        // TODO validate argument at position index
        const validationFailed = false
        if (validationFailed) { return { statusCode: StatusCodes.BadInvalidArgument }}
    })

    // initiate program run
    const deviceProgramRunId = `Run-${++runId}`
    // run program async
    runProgram(deviceProgramRunId, inputArguments)
    // return run-Id
    return {
        outputArguments: [new Variant({ dataType: DataType.String, value: deviceProgramRunId })],
        statusCode: StatusCodes.Good
    }
}
```

Access active-program, result-set & state-machine of functional-unit

Bind function to StartProgram(call)

Validate current state of FunctionalUnit
If not ready return BadInvalidState

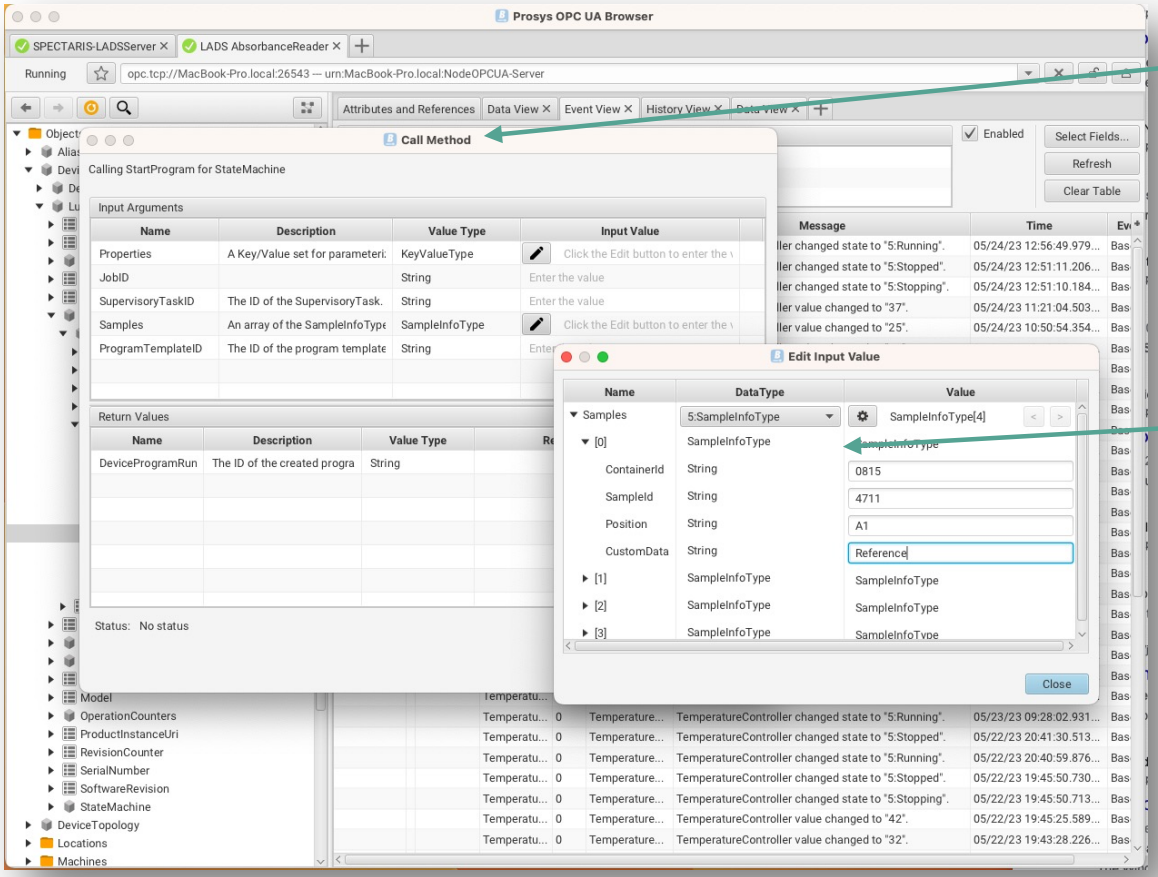
Validate provided InputArguments
If not good return BadInvalidArgument

If “good to go” increase the run-id, call async runProgram(inputArguments), and return the new run-id to the client

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Can I provide job-id, task-id, samples, .. when starting programs?



Yes, yes, yes, ..

For each sample you can provide,
its container-id,
its sample-id,
its position,
your custom-data

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Executing program-runs..

```
async function runProgram(deviceProgramRunId: string, inputArguments: VariantLike[]) {
    // dynamically create an new result object in the result set
    const resultType = getLADSObjectType(addressSpace, "ResultType")
    const result = <LADSResult>resultType.instantiate({ componentOf: <BaseNode><unknown>resultSet, browseName: deviceProgramRunId })
    // get program template-id
    const programTemplateId: string = inputArguments[0].value
    // set context information provided by input-arguments
    result.properties?.setValueFromSource(inputArguments[1])
    result.jobId?.setValueFromSource(inputArguments[2])
    result.supervisoryTaskId?.setValueFromSource(inputArguments[3])
    result.samples?.setValueFromSource(inputArguments[4])
    result.started?.setValueFromSource({ dataType: DataType.DateTime, value: new Date() })
    // initialize active-program runtime properties
    const runTime = 30000 //ms
    const delta = 500 //ms
    activeProgram.currentRuntime?.setValueFromSource({ dataType: DataType.Double, value: 0 })
    activeProgram.currentStepName?.setValueFromSource({ dataType: DataType.LocalizedText, value: 'Measure' })
    activeProgram.currentStepNumber?.setValueFromSource({ dataType: DataType.UInt32, value: 1 })
    activeProgram.currentStepRuntime?.setValueFromSource({ dataType: DataType.Double, value: 0 })
    activeProgram.estimatedRuntime?.setValueFromSource({ dataType: DataType.Double, value: runTime })
    activeProgram.deviceProgramRunId?.setValueFromSource({ dataType: DataType.String, value: deviceProgramRunId })
    // start all required functions
    functionalUnitStateMachine.setState("Running")
    temperatureControllerStateMachine.setState("Running")
    for (let t = 0; t <= runTime; t += delta) {
        // update active-program runtime properties
        activeProgram.currentRuntime?.setValueFromSource({ dataType: DataType.Double, value: t })
        activeProgram.currentStepRuntime?.setValueFromSource({ dataType: DataType.Double, value: t })
        // do whatever is necessary for the run
        await sleepMilliSeconds(delta)
    }
```

Create a new result in the result-set

Get and load (omitted in this example) the ProgramTemplate

Copy context information provided by StartProgram() to the result and document start timestamp

Prepare runtime properties

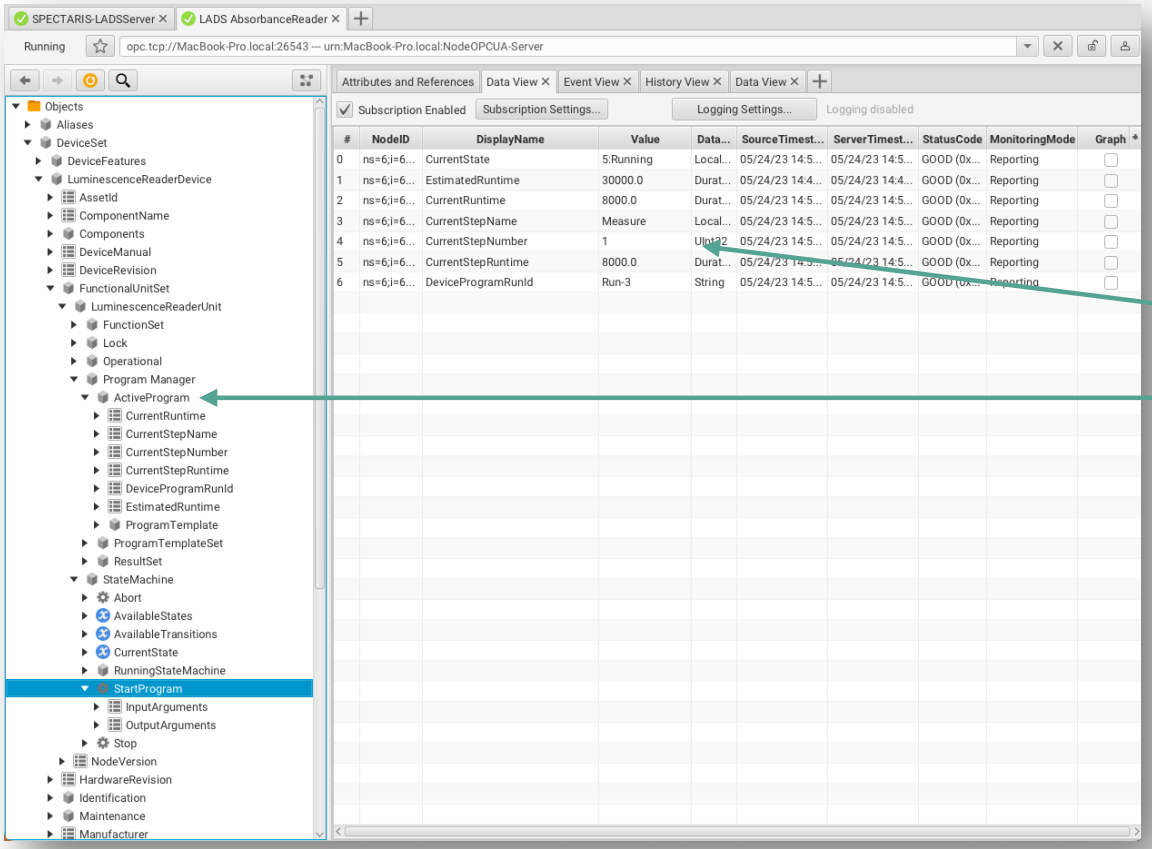
Start all required functions

Update runtime properties and execute run until finished

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Monitoring the program-run's progress ..



Monitor run in real-time

Active-Program exposes runtime properties

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Finalizing program-runs and providing result data..

```
temperatureControllerStateMachine.setState("Stopped")
functionalUnitStateMachine.setState("Stopped")
// finalize
result.stopped?.setValueFromSource({ dataType: DataType.DateTime, value: new Date() })
activeProgram.currentStepName?.setValueFromSource({ dataType: DataType.LocalizedText, value: 'Finished' })
activeProgram.currentStepNumber?.setValueFromSource({ dataType: DataType.UInt32, value: 2 })
// get luminescence readings and add to VariableSet in result
const readings = luminescenceSensor.sensorValue.readValue()
result.namespace.addVariable({
    propertyOf: result.variableSet,
    browseName: 'Luminescence',
    dataType: DataType.Double,
    valueRank: 1,
    arrayDimensions: [wells],
    value: readings.value
})
```

Stop all required functionality

Document stop timestamp and report finished

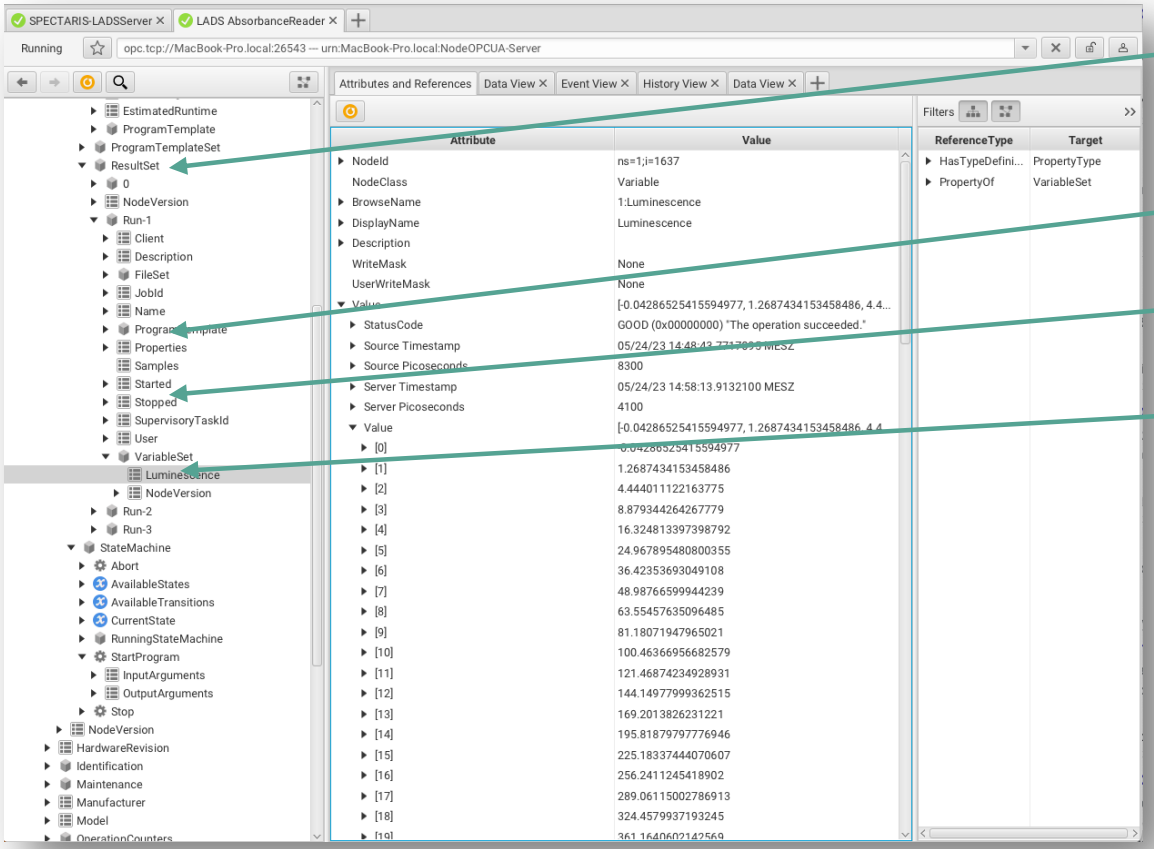
Get important data like variables, structures, files

Document result data in variable-set or file-set

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Accessing program-run results ..



Result-set with results of stored runs

Meta data of Run-1

Timestamps of Run-1

Endpoints/variables of Run-1:
luminescence readings with array[96]

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

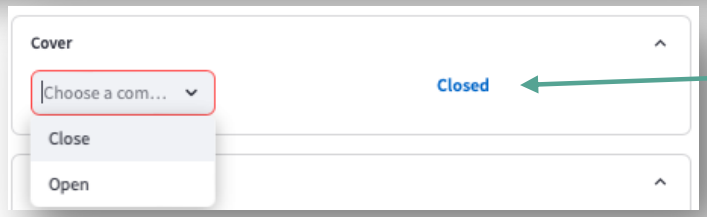
Step 10:
Implementing the Cover State-Machine

State-machines, transitions, methods, events, ..?

Implementing the Cover State-Machine

```
//-------------------------------------------------------------
// Step 10: Implemement cover state-machine
//-------------------------------------------------------------
const cover = functionSet.cover
const coverState = cover.coverState
const coverStateMachine = promoteToFiniteStateMachine(coverState)
coverStateMachine.setState(LADSCoverState.Closed)
coverState.open?.bindMethod(transite.bind(coverStateMachine, LADSCoverState.Closed, LADSCoverState.Opened, c
coverState.close?.bindMethod(transite.bind(coverStateMachine, LADSCoverState.Open
coverState.lock?.bindMethod(transite.bind(coverStateMachine, LADSCoverState.Closed, LADSCoverState.Locked, c
coverState.unlock?.bindMethod(transite.bind(coverStateMachine, LADSCoverState.Locked, LADSCoverState.Closed,

async function transite(this: UAStateMachineEx, fromStateName: string, toStateName: string, eventSource: UAOb
    const state = this.getCurrentState()
    if (state == null || state?.includes(fromStateName)) {
        this.setState(toStateName)
        if (eventSource && baseEventType)
            eventSource.raiseEvent(baseEventType, { message: { dataType: DataType.LocalizedText, value: `${ev
        return { statusCode: StatusCodes.Good}
    } else {
        return {statusCode: StatusCodes.BadInvalidState}
    }
}
```



Get state-machine object and promote to finite state-machine

Bind all required methods to one function

The function transite() checks validity of states, does the transition, raises event, ..

Monitor & control from remote with LADS OPC UA client ..

Step 11:
Implement a Discrete Control-Function

There is more than analog controllers – discrete controllers e.g., :

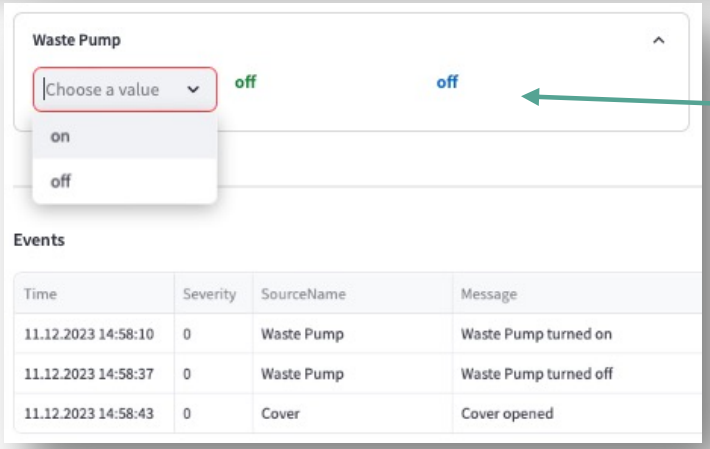
- TwoStateDiscreteControlFunction for simple valves, ..
- MultiStateDiscreteControlFunction for HPLC valves, ..

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Implementing a Discrete ControlFunction

```
//-------------------------------------------------------------
// Step 11: Implemement waste-pump as two-state control-function
//-------------------------------------------------------------
const wastePump = functionSet.wastePump
promoteToFiniteStateMachine(wastePump.controlFunctionState).setState("Running")
const wastePumpTrueStateName = wastePump.targetValue.trueState.readValue().value.value.text
const wastePumpFalseStateName = wastePump.targetValue.falseState.readValue().value.value.text
wastePump.targetValue.on("value_changed", (dataValue: DataValueT<boolean, DataType.Boolean>) => {
    wastePump.currentValue.setValueFromSource(dataValue.value)
    const valueName = dataValue.value.value?wastePumpTrueStateName:wastePumpFalseStateName
    wastePump.raiseEvent(baseEventType, { message: { dataType: DataType.LocalizedText, value: `${wastePump.getD
})
```



Get state-machine object, promote to state-machine and turn the control-function on

Fetch the human readable names of the specific discrete values

Bind a simple event-handler which performs the control action, provides feed-back and raises an event message

Monitor & control from remote with LADS OPC UA client ..

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 12:
Simply add enhanced features to your LADS Device

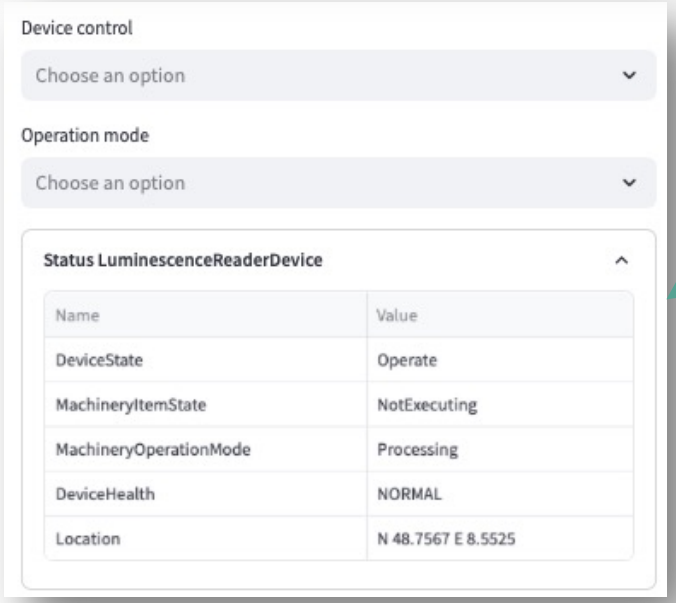
One line of code adds enhanced features

- Event monitoring with specific scope (function, functional-unit, device, ..)
- Compatibility with Machinery Basics Companion Specification
- Default Device state-machines behavior in accordance with OPC 30500-1 annex B

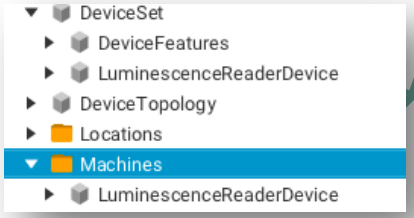
DeviceHelper Magic..

```
//-------------------------------------------------------------
// Step 12: Attach LADSDeviceHelper to implememt standard behavior
// for event propagation and device level state-machines
//-------------------------------------------------------------
const deviceHelper = new LADSDeviceHelper(luminescenceReaderDevice, {initializationTime: 2000, shutdownTime: 2000, raiseEvents: true})
```

Simply add this line of code to ..



.. to automatically provide the state-machine behavior in accordance with LADS Annex B and "Industry 4.0" Machinery Basics spec.



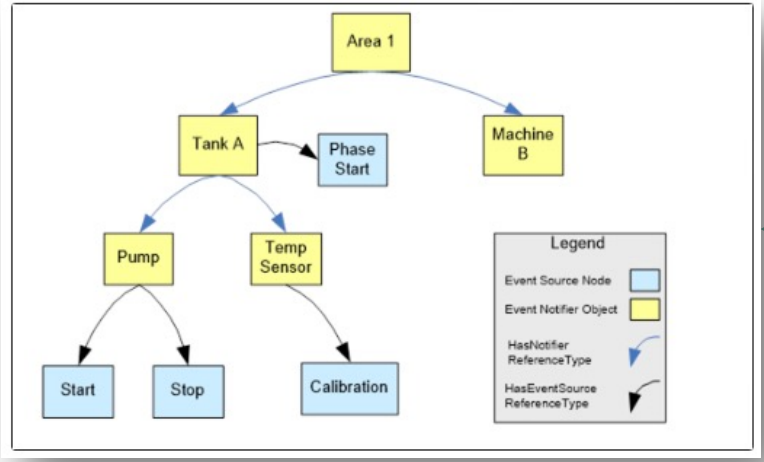
LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

DeviceHelper Magic..

```
//-------------------------------------------------------------------
// Step 12: Attach LADSDeviceHelper to implememt standard behavior
// for event propagation and device level state-machines
//-------------------------------------------------------------------
const deviceHelper = new LADSDeviceHelper(luminescenceReaderDevice, {initializationTime: 2000, shutdownTime: 2000, raiseEvents: true})
```

Simply add this line of code to ..



.. automatically create event scopes
for Device, Function & FunctionalUnit objects,
which ease subscribing to events at specific levels
(see OPC UA Part 3, 7.18)

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Step 13:
Enter the World of Process Control (TagVariables, Alias, ..)

One line of code automatically adds a flat list of TagVariables representing essential information of your device..

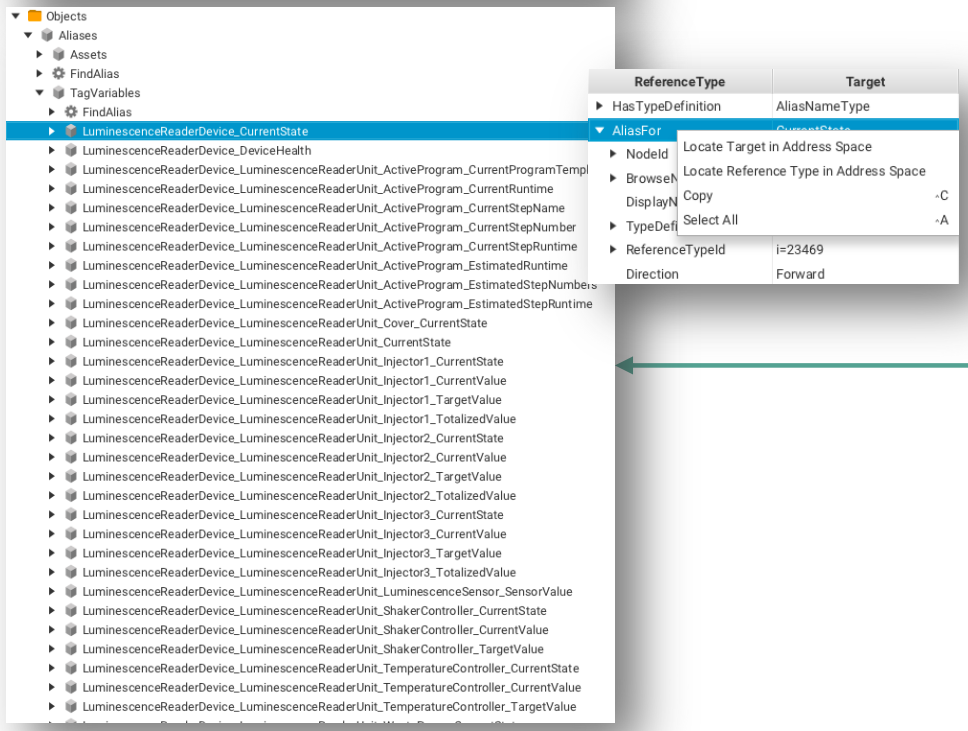
LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY in the German Industry Association SPECTARIS

TagVariables Magic..

```
//-----------------------------------------------------------------------
// Step 13: Create Tag-Variable aliases
//-----------------------------------------------------------------------
addAliases(luminescenceReaderDevice)
```

Simply add this line of code to ..



.. automatically create la list of TagVariables representing essential information in accordance with OPC UA Part 17: Alias Names

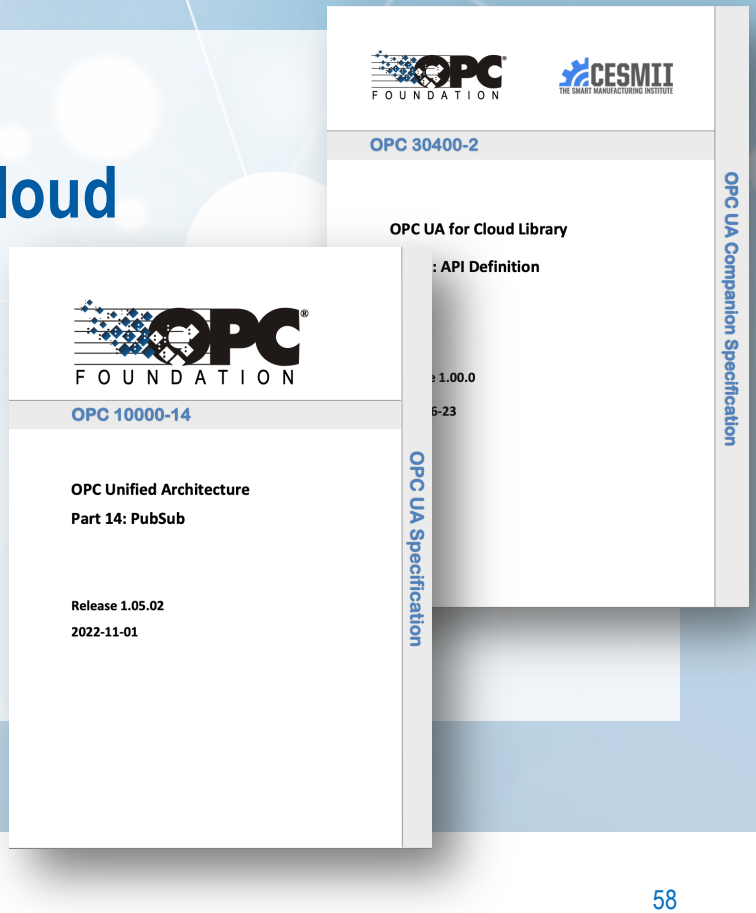
LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Bonus feature - LADS OPC UA in the Cloud

Unleashing the power of the OPC UA eco-system!

- PubSub
- Cloud Library

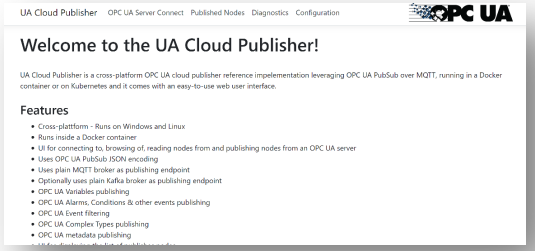


LADS
Laboratory & Analytical Device Standard

ANALYTICAL, BIO AND LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

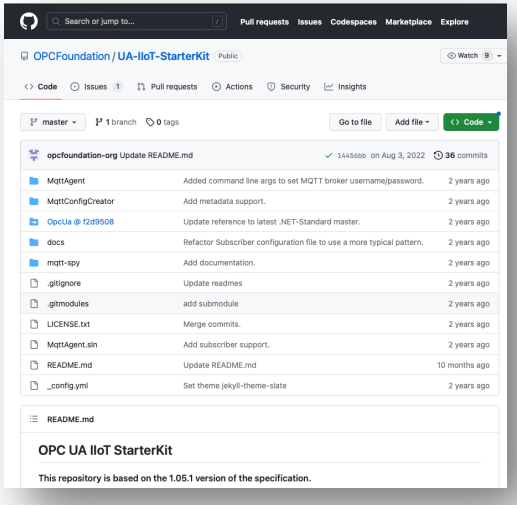
Publishing selected data via OPC UA PubSub to Cloud

Connect via an Edge Gateway e.g., UA Cloud Publisher



https://github.com/barnstee/UA-CloudPublisher

Directly integrate OPC UA PubSub into your device



LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Bonus Feature - Robots & Cobots welcome!

Unleashing the power of the OPC UA eco-system!

- Robotics
- Relative Spatial Locations

LADS
Laboratory & Analytical
Device Standard

ANALYTICAL, BIO AND
LABORATORY TECHNOLOGY
in the German Industry Association
SPECTARIS

Introducing the virtual cobot-enabled LADS OPC UA laboratory