SESKO – Iot and Digital Twin Workshop OPC UA and FMU Digital Twin

D.Sc. Mika Karaila – mika.karaila@valmet.com

Tuomas Laine (Master of science worker) – tuomas.laine@valmet.com

Part of ”Virtual Sea Trial” project - https://virtualseatrial.fi/

Standards will enable interoperability

OPC UA:

- Standardized data models freely available for over 60 types of industrial equipment, published by the OPC Foundation via Companion Specifications
- Extensible security profiles, including authentication, authorization, encryption and checksums
- Extensible security key management, including X.509, token and password
- Support for both client-server and publish-subscribe communication patterns
- Communication protocol independent. Mappings to several communication protocols like TCP/IP, UDP/IP, WebSockets, AMQP and MQTT are specified
- Initially successful in standardized data exchange with industrial equipment (discrete manufacturing, process manufacturing, energy) and systems for data collection and control, but now also leveraged in building automation, weighing and kitchen equipment and cloud applications
- Open – open-source reference implementations freely available to OPC Foundation members, non members under GPL 2.0 license[2]
- Cross-platform – not tied to one operating system or programming language
- Service-oriented architecture (SOA)
- The specification is freely available on the OPC Foundation website and is split into several parts to ease implementation, but only OPC UA stack vendors need to read them, end users simply leverage existing commercial and/or open-source stacks available in all popular programming languages

FMI/FMU:

- The Functional Mock-up Interface (or FMI) defines a standardized interface to be used in computer simulations to develop complex cyber-physical systems.
- The vision of FMI is to support this approach: if the real product is to be assembled from a wide range of parts interacting in complex ways, each controlled by a complex set of physical laws, then it should be possible to create a virtual product that can be assembled from a set of models that each represent a combination of parts, each a model of the physical laws as well as a model of the control systems (using electronics, hydraulics, and digital software) assembled digitally. The FMI standard thus provides the means for model based development of systems and is used for example for designing functions that are driven by electronic devices inside vehicles (e.g. ESP controllers, active safety systems, combustion controllers). Activities from systems modelling, simulation, validation and test can be covered with the FMI based approach.
- The four required FMI aspects of creating models capable of being assembled have been covered in Modelisar project:
  - FMI for model exchange,
  - FMI for co-simulation,
  - FMI for applications,
  - FMI for PLM (integration of models and related data in product life-cycle management).
- In practice, the FMI implementation by a software modelling tool enables the creation of a simulation model that can be interconnected or the creation of a software library called FMU (Functional Mock-up Unit)

OPC UA latest releases

2019

Releases
OPC 10000-100, UA for Devices, Version 1.02
OPC 40010-1, Robotics Part 1
OPC 30070-1, MTConnect Part 1
OPC 30200, Commercial Kitchen Equipment
OPC 40100-1, Machine Vision Companion Specification

New Working Groups
CIP Devices by ODVA
Mining Machinery by VDMA
Weihenstephan standards by Weihenstephan Standards, VDMA Food
ISA 100 by ISA 100 Wireless Compliance Institute (WCI)
Pumps and Vacuum Pumps by VDMA
Harmonization by OPC Foundation
Glass Industries by VDMA
Plastics and Rubber Machinery by VDMA
Surface Technology by VDMA
Woodworking Machinery by VDMA

StartOPC Hub India, Singapore/ASEAN, France

OPC Foundation membership
737 members (end 2019)

2020

Releases
OPC 30000 – IEC 61131-3, Version 1.2 PLCOpen
OPC 30260 – OpenSCS
OPC 30050 – PackML
OPC UA, Ellyptic Curve Cryptography
OPC 40501-2, UA for MachineTools, Part 1: Base Model
OPC 40001-1, UA for Machinery, Part 1: Basic Building Blocks
OPC 10000-200, UA for Industrial Automation
OPC 30010 – UA CS for AutoID, 1.01.07
OPC 40200, UA for Weighing Technology
OPC 30081, UA for Process Automation – PADIM

New Working Groups
Cranes and Hoists by VDMA
Length Measuring Systems (LMS) by VDMA
ISA-95 Version 2 by OPC Foundation
Fibre and Yarn Testing Devices (FYTD) by VDMA
Process Air Extraction and Filtration Systems – PAEFS by VDMA
M2X Intralogistics Communication by VDMA
Compressed Air Systems by VDMA
Cloud Library by CESMII,
LADS – Laboratory and Analytical Device Standard by Spectaris
Machinery Common Model by VDMA

OPC Foundation membership
803 members (end 2020)

2021

Releases
OPC 40001-1, UA for Machinery, Part 1: Basic Build. Blocks – 1.01
OPC 30141 – PROFIenergy
OPC 40084-12 – Extrusion – Calender
OPC 10000-100 – OPC UA for Devices, v1.03
OPC 11031-4 – ISA-95 Job Order OPC 30261 – OpenSCS Job Orders
OPC 40223 – Pumps and Vacuum Pumps
OPC 40083 – PlasticsRubber – General Types
OPC 30270 – UA for Asset Administration Shell (AAS)
OPC 40250-1, UA for Compressed Air Systems
OPC 10000-200, UA for Industrial Automation, v1.01
OPC 40600 – Weihenstephan Standards

New Working Groups
ISA-95 Version 2
Fibre and Yarn Testing Devices (FYTD)
Process Air Extraction and Filtration Systems – PAEFS
LADS – Laboratory and Analytical Device Standard
M2X Intralogistics Communication
Cranes and Hoists
Length Measuring Systems (LMS)
Cloud Federation

OPC Foundation membership
840 members (Aug 2021)

2022

Releases

- OPC 10000-1..24, OPC UA v1.05
- OPC 10000-100, Device Model v1.03
- OPC 10000-110, Asset Mgmt Basics
- OPC 10000-200, Industrial Automation v1.01
- OPC 30400-1..2, Cloud Library
- OPC 30080, FDI v1.03
- OPC 30142, PROFINET-RemoteIO
- OPC 40001-1 Machinery Basic Building Blocks v1.02
- OPC 40501-1 Machine Tools v1.01
- OPC 40084-1..12 PlasticsRubber-Extrusion v2.00
- OPC 30060 Tobacco Machinery v2.00
- OPC 4056n Mining

New Working Groups

- Power Consumption Mgmt
- Additive Manufacturing
- Analyzer System Integration
- Global Positioning

Foundation membership
895 members (Sept 2022)

FMI: Functional Mock-up Interface

Overview

- The Functional Mock-up Interface (FMI) is a free standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code, distributed as a ZIP file. It is supported by more than 170 tools and maintained as a Modelica Association Project (MAP FMI). Releases and issues can be found on github.com/modelica/fmi-standard.
- The Functional Mock-up Interface (FMI) defines a ZIP archive and an application programming interface (API) to exchange dynamic models using a combination of XML files, binaries and C code: the Functional Mock-up Unit (FMU). The API is used by a simulation environment, the importer, to create one or more instances of an FMU and to simulate them, typically together with other models. The FMI defines three interface types:
  - Co-Simulation (CS) where the FMU typically contains its own solver or scheduler,
  - Model Exchange (ME) that requires the importer to perform numerical integration, and
  - Scheduled Execution (SE) where the importer triggers the execution of the model partitions.
- This document does not describe how to generate an FMU from a modeling environment.
- The interface types have large parts in common, defined in Common Concepts. In particular:
  - FMI Application Programming Interface (C) — Section 2.2
  - All required computations are triggered by calling standardized C functions from the importer into the FMU. The FMU can signal certain events back to the importer using callback functions provided by the importer. C is used because it is the most portable programming language today and is the only programming language that can be utilized in all embedded control systems. The FMI API does not restrict what operating system services an FMU can use on the platform it runs on. However, for maximal portability, any dependency on the target platform should be minimized and operating system services should be accessed only through standard libraries. Special run-time requirements should be documented in the appropriate directory inside the ZIP file.
  - FMI Description Schema (XML)
  - The schema defines the structure and content of a model description file (modelDescription.xml), for example, generated by a modeling environment. This XML file contains the definition of all exposed variables, their interdependencies (model structure) and capability flags of the FMU. Providing variable descriptions outside the C-API allows an importer to access and store the variable definitions (without any memory or efficiency overhead of standardized access functions) in its own representation.
  - FMU Distribution (ZIP) — Section 2.5
  - An FMU is distributed as one ZIP file. The ZIP file contains the FMI modelDescription.xml, the binaries and libraries required to execute the FMI functions (.dll or .so files), and/or the sources of the FMI functions, documentation, and other data used by the FMU (e.g., tables or maps).

FMI: Two types of simulation models

1.2.1. FMI for Model Exchange (ME)

The Model Exchange interface exposes an ODE to an external solver of an importer. Models are described by differential, algebraic and discrete equations with time-, state- and step-events. That integration algorithm of the importer, usually a ODE/DAE solver, is responsible for advancing time, setting states, handling events, etc. (See Section 3.)

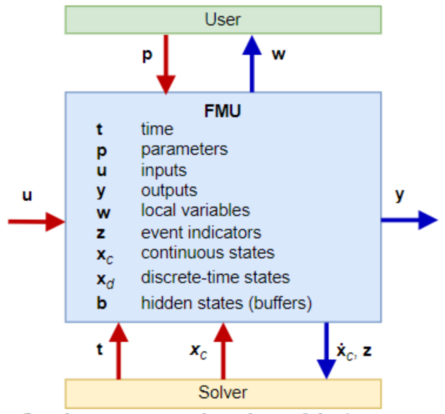


Figure 1. Schematic view of data flow between user, the solver of the importer and the FMU for Model Exchange

1.2.2. FMI for Co-Simulation (CS)

The Co-Simulation interface is designed both for the coupling of simulation tools, and the coupling of subsystem models, exported by a modeling environment together with their solvers as runnable code. (See Section 4.)

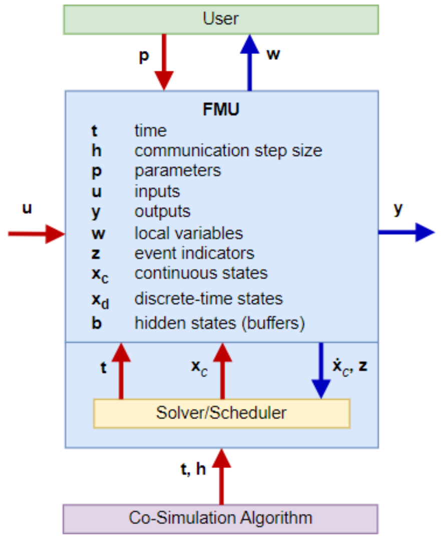
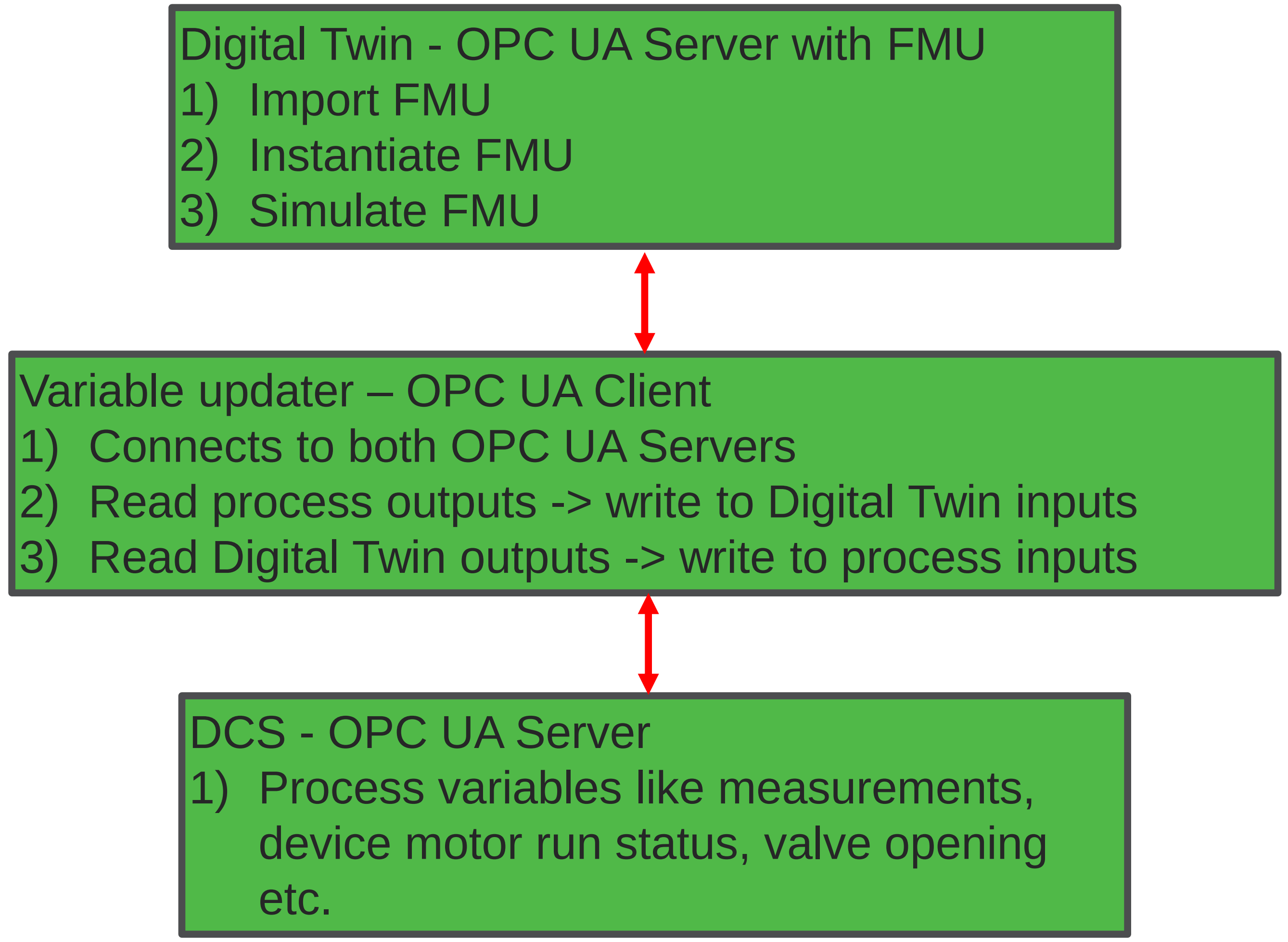


Figure 2. Schematic view of data flow between user, the co-simulation algorithm of the importer and the FMU for Co-Simulation

Combining OPC UA and FMU together

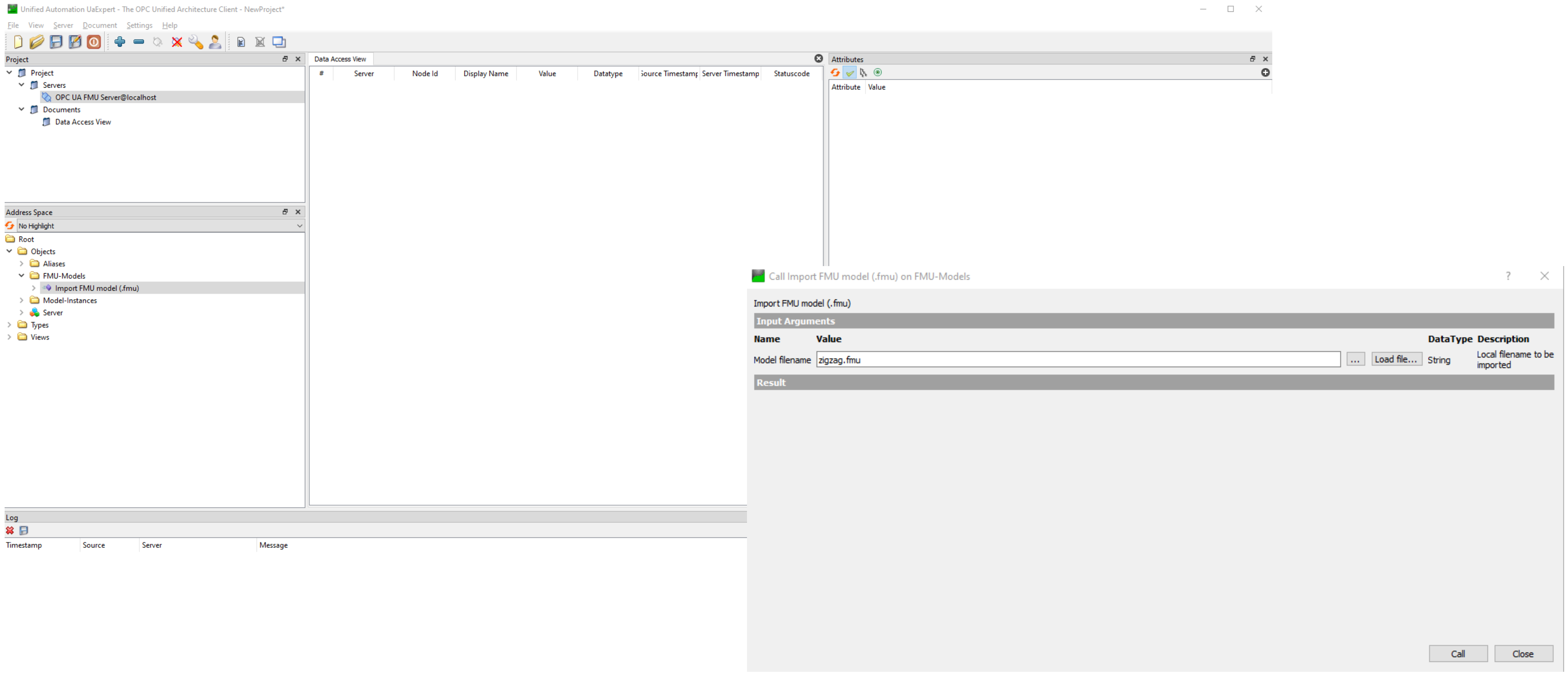
Basic idea


Digital Twin - OPC UA Server with FMU
1) Import FMU
2) Instantiate FMU
3) Simulate FMU
Variable updater – OPC UA Client
1) Connects to both OPC UA Servers
2) Read process outputs -> write to Digital Twin inputs
3) Read Digital Twin outputs -> write to process inputs
DCS - OPC UA Server
1) Process variables like measurements, device motor run status, valve opening etc.

Implemented proto-type

OPC UA FMU Server

Server started, nothing imported => Run method to import new FMU file


Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File View Server Document Settings Help
Project
Project
Servers
OPC UA FMU Server@localhost
Documents
Data Access View
Address Space
No Highlight
Root
Objects
Aliases
FMU-Models
Import FMU model (.fmu)
Model-Instances
Server
Types
Views
Data Access View
# Server Node Id Display Name Value Datatype Source Timestamp Server Timestamp Statuscode
Attributes
Attribute Value
Log
Timestamp Source Server Message
Call
Close

Call Import FMU model (.fmu) on FMU-Models

Import FMU model (.fmu)

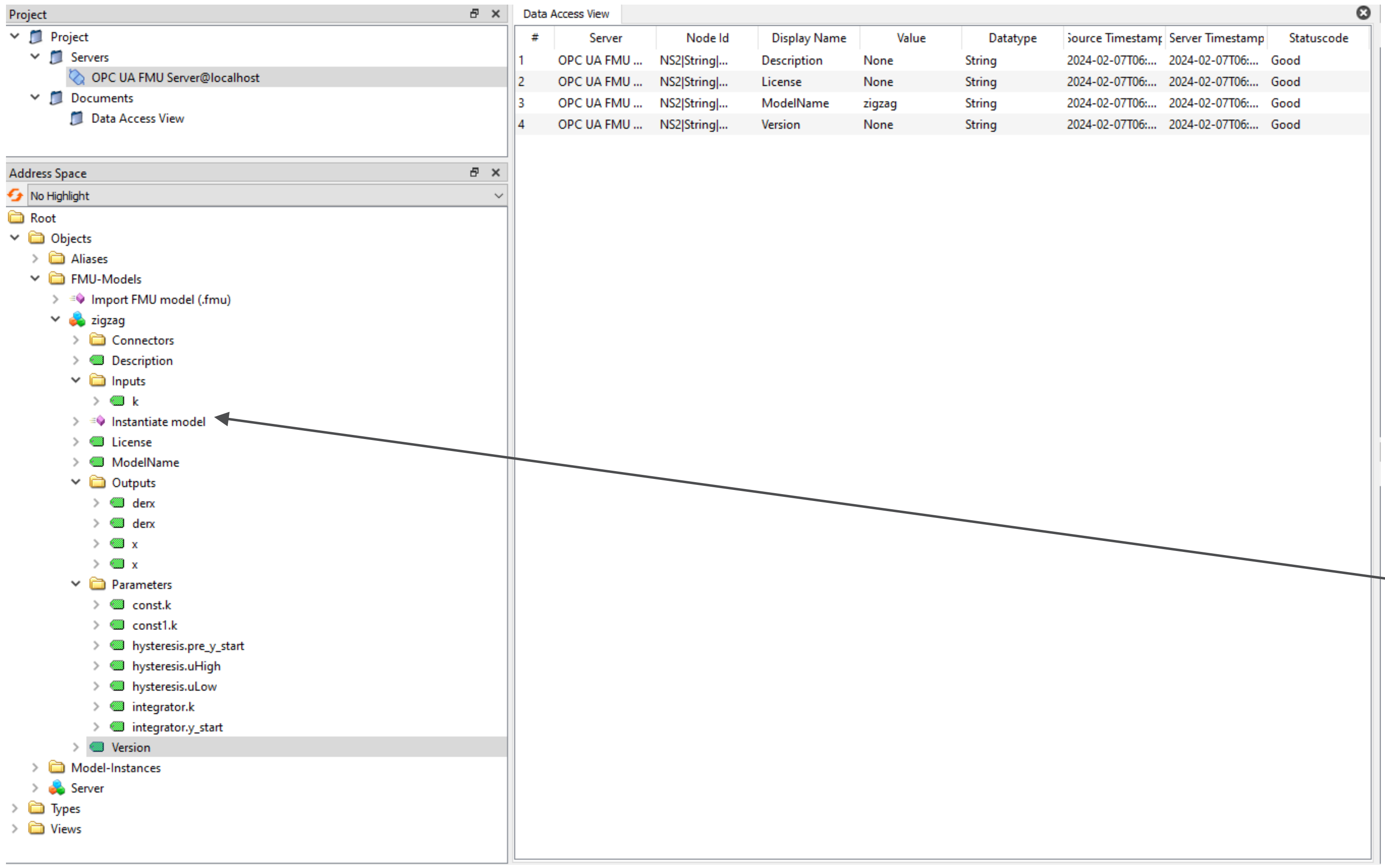
Input Arguments

Name	Value	DataType	Description
Model filename	zigzag.fmu	String	Local filename to be imported

Result

OPC UA FMU Server

FMU model metadata and all inputs, outputs, parameters created into the address space



Project
- Project
  - Servers
    - OPC UA FMU Server@localhost
  - Documents
    - Data Access View

Address Space

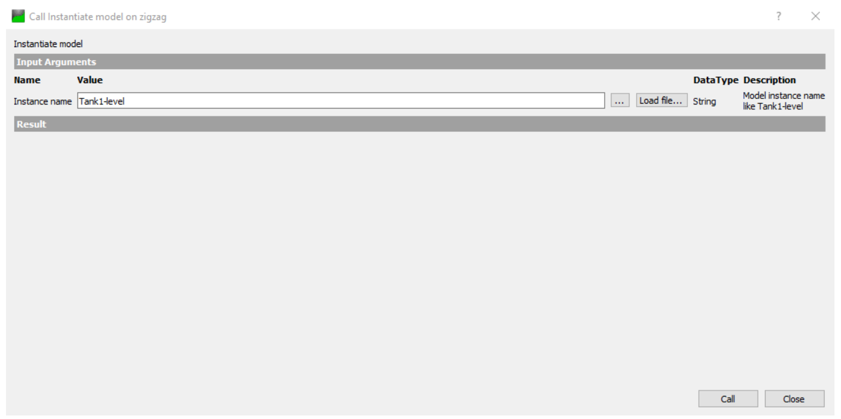
No Highlight

- Root
- Objects
  - Aliases
  - FMU-Models
    - Import FMU model (.fmu)
    - zigzag
      - Connectors
      - Description
      - Inputs
        - k
      - Instantiate model
      - License
      - ModelName
      - Outputs
        - derx
        - derx
        - x
        - x
      - Parameters
        - const.k
        - const1.k
        - hysteresis.pre_y_start
        - hysteresis.uHigh
        - hysteresis.uLow
        - integrator.k
        - integrator.y_start
      - Version
  - Model-Instances
  - Server
- Types
- Views

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	OPC UA FMU ...	NS2|String|...	Description	None	String	2024-02-07T06:...	2024-02-07T06:...	Good
2	OPC UA FMU ...	NS2|String|...	License	None	String	2024-02-07T06:...	2024-02-07T06:...	Good
3	OPC UA FMU ...	NS2|String|...	ModelName	zigzag	String	2024-02-07T06:...	2024-02-07T06:...	Good
4	OPC UA FMU ...	NS2|String|...	Version	None	String	2024-02-07T06:...	2024-02-07T06:...	Good

Instantiate must be called to instantiate model



Call Instantiate model on zigzag

Instantiate model

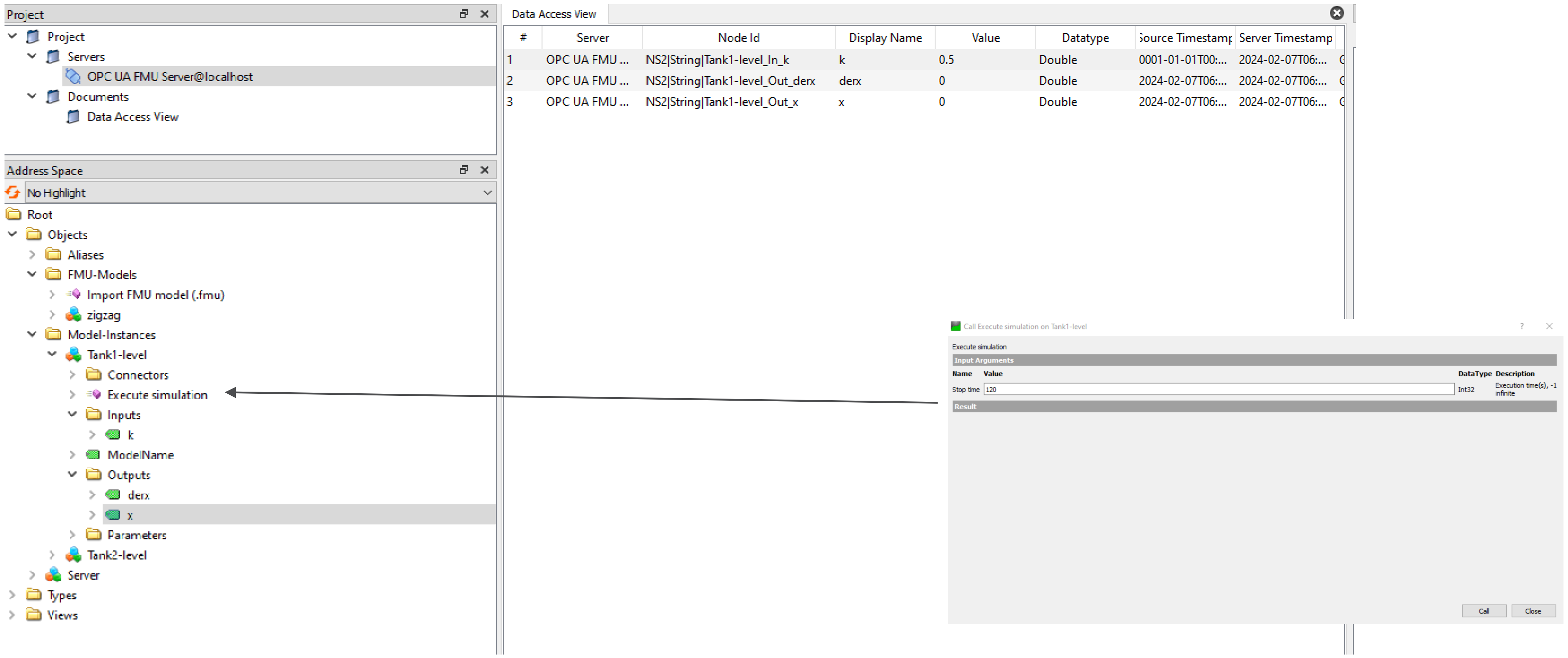
Input Arguments

Name	Value	DataType	Description
Instance name	Tank1-level	String	Model instance name like Tank1-level

Result

OPC UA FMU Server

FMU model instance, modify needed variable(s) and call Execute simulation



#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp
1	OPC UA FMU ...	NS2|String|Tank1-level_In_k	k	0.5	Double	0001-01-01T00:...	2024-02-07T06:...
2	OPC UA FMU ...	NS2|String|Tank1-level_Out_derx	derx	0	Double	2024-02-07T06:...	2024-02-07T06:...
3	OPC UA FMU ...	NS2|String|Tank1-level_Out_x	x	0	Double	2024-02-07T06:...	2024-02-07T06:...

Call Execute simulation on Tank1-level

Execute simulation

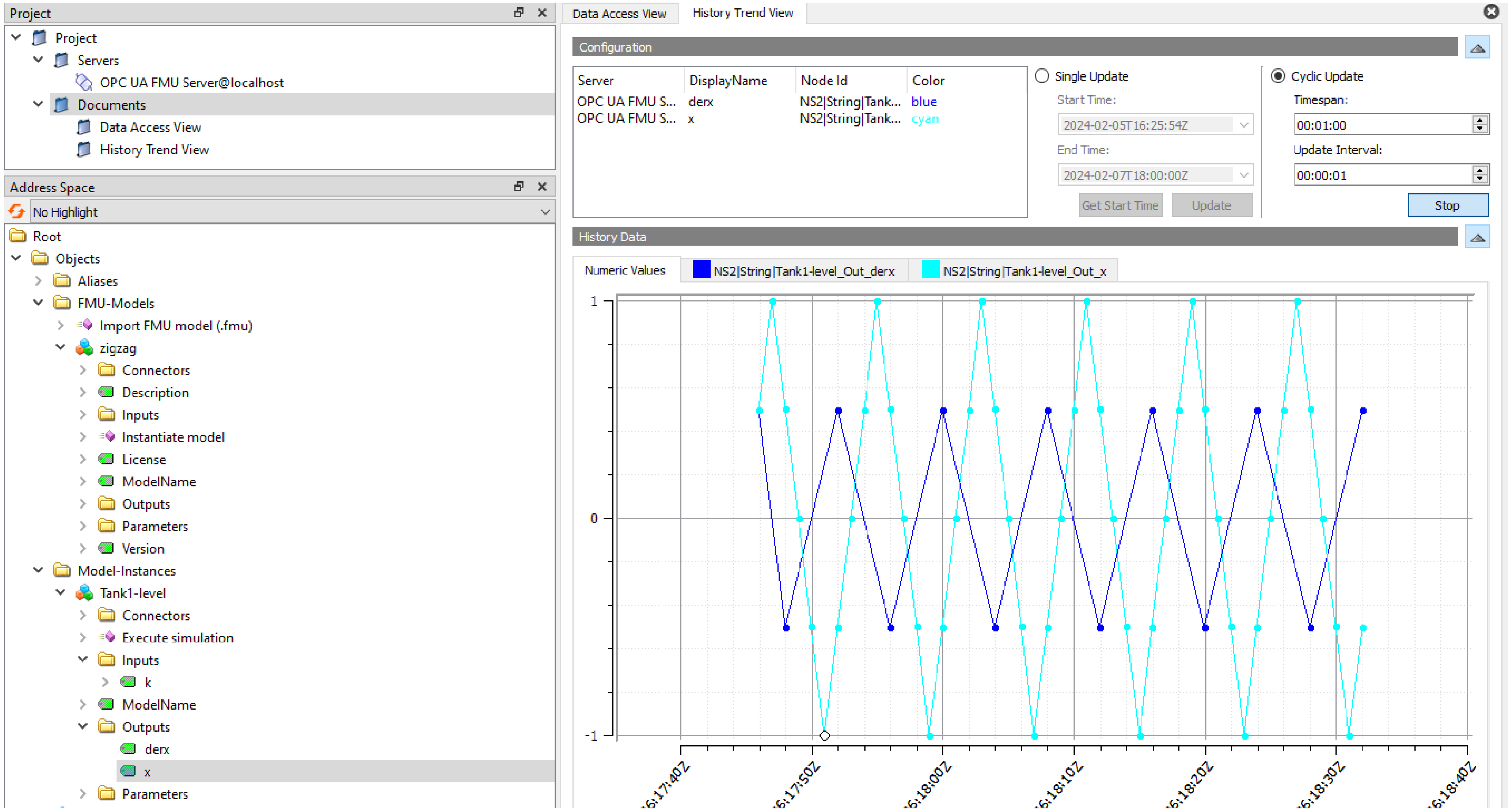
Input Arguments

Name	Value	DataType	Description
Stop time	120	Int32	Execution time(s), -1 infinite

Result

OPC UA FMU Server

FMU model exchange in execution, OPC UA variables can be shown in trend (HA)



Project
- Project
  - Servers
    - OPC UA FMU Server@localhost
  - Documents
    - Data Access View
    - History Trend View

Address Space — No Highlight
- Root
  - Objects
    - Aliases
    - FMU-Models
      - Import FMU model (.fmu)
      - zigzag
        - Connectors
        - Description
        - Inputs
        - Instantiate model
        - License
        - ModelName
        - Outputs
        - Parameters
        - Version
    - Model-Instances
      - Tank1-level
        - Connectors
        - Execute simulation
        - Inputs
          - k
        - ModelName
        - Outputs
          - derx
          - x
        - Parameters

Data Access View | History Trend View

Configuration

Server	DisplayName	Node Id	Color
OPC UA FMU S...	derx	NS2|String|Tank...	blue
OPC UA FMU S...	x	NS2|String|Tank...	cyan

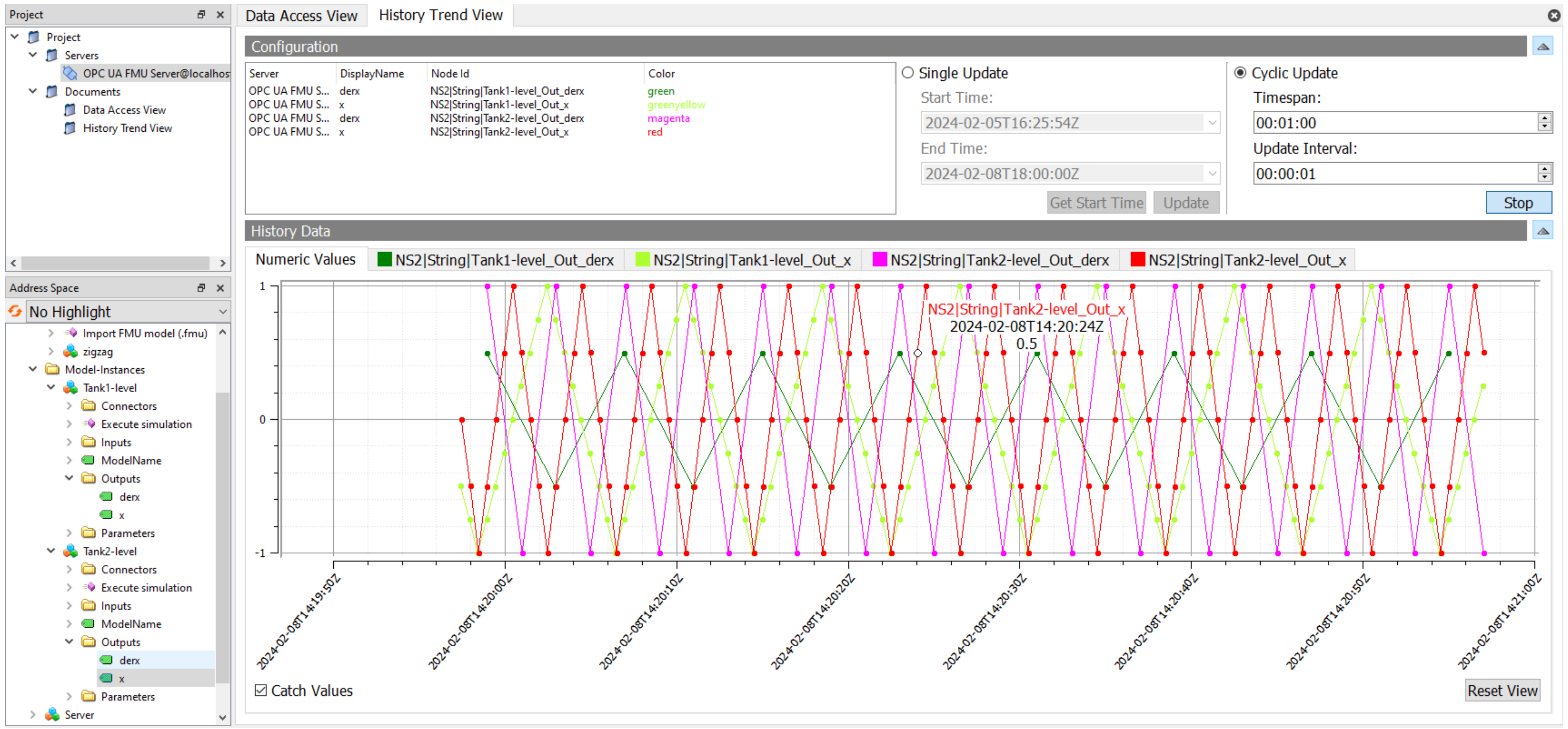
Single Update — Start Time: 2024-02-05T16:25:54Z — End Time: 2024-02-07T18:00:00Z — Get Start Time — Update

[✓ Cyclic Update] — Timespan: 00:01:00 — Update Interval: 00:00:01 — Stop

History Data

Numeric Values | NS2|String|Tank1-level_Out_derx | NS2|String|Tank1-level_Out_x

Running parallel two “tank” level simulations



Server	DisplayName	Node Id	Color
OPC UA FMU S...	derx	NS2|String|Tank1-level_Out_derx	green
OPC UA FMU S...	x	NS2|String|Tank1-level_Out_x	greenyellow
OPC UA FMU S...	derx	NS2|String|Tank2-level_Out_derx	magenta
OPC UA FMU S...	x	NS2|String|Tank2-level_Out_x	red

○ Single Update
Start Time: 2024-02-05T16:25:54Z
End Time: 2024-02-08T18:00:00Z

● Cyclic Update
Timespan: 00:01:00
Update Interval: 00:00:01

☑ Catch Values

Summary

- Needs more work and testing for scalability etc.
  - More features implemented like history option added for the input/output variables
- More FMU models to be tested to find out missing features and errors
  - Co-Simulation implemented & tested (Tuomas)
  - More FMU models needed to cover that implementation is sufficient & robust
- Connection for DCS with OPC UA Client for FAT testing should be “easy”
  - Matching names mapped & connected, browse and generate link DCS <-> OPC UA Client <-> OPC UA Server

Final conclusion:

Promising implementation that will give more flexibility & testing capabilities

Valmet

FORWARD