HANDBOOK
Python Powerfactory OPC-UA

Python project
https://git.rwth-aachen.de/acs/research/N5GEH/tud/TUD_usecases/tree/master/UC2_grid_protection/CloudSetup

- look for README to get install advices
- look for setup.py to get dependencies

Install the free OPC-UA Client from UA Expert
https://www.unified-automation.com/de/produkte/entwicklerwerkzeuge/uaexpert.html

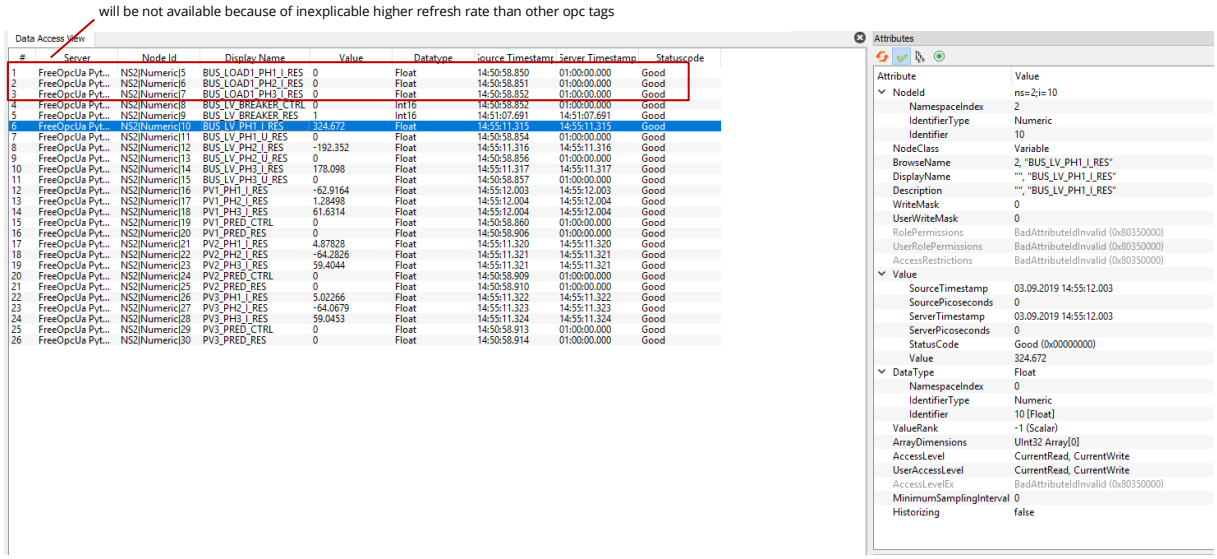

Bild 1: List of variables @ OPC-UA server; right: attributes of selected variable named „BUS_LV_PH1_I_RES" (relevant is SourceTimestamp, Value, DataType and for connection with OPC-UA the NodeId)

You can change the values of each node by simple typing in new ones.

Setup PowerFactory

- go to Control panel "Help" --> "More Components" --> OPC-API to open help document to setup your PowerFactory to used it with OPC-UA
- Maybe you have to switch the graphic acceleration to software under "tools" – "configuration" – "extended" – "extended"
- Import project UC_Grid_Protection.pfd and activate
- you will find a simple LV grid with predefined external measurement devices prepared as OPC–UA nodes (cf. Bild 2)
- after you set up Python stuff (cf. Run main.py), execute (1) and check console for success
- start simulation by (2) and (3) at Bild 4 and p.r.n. minimize the PowerFactory window to let OPC-UA-Link work properly
- when simulation stopped, finished by time, … check (4) and delete newly created events before start simulation again

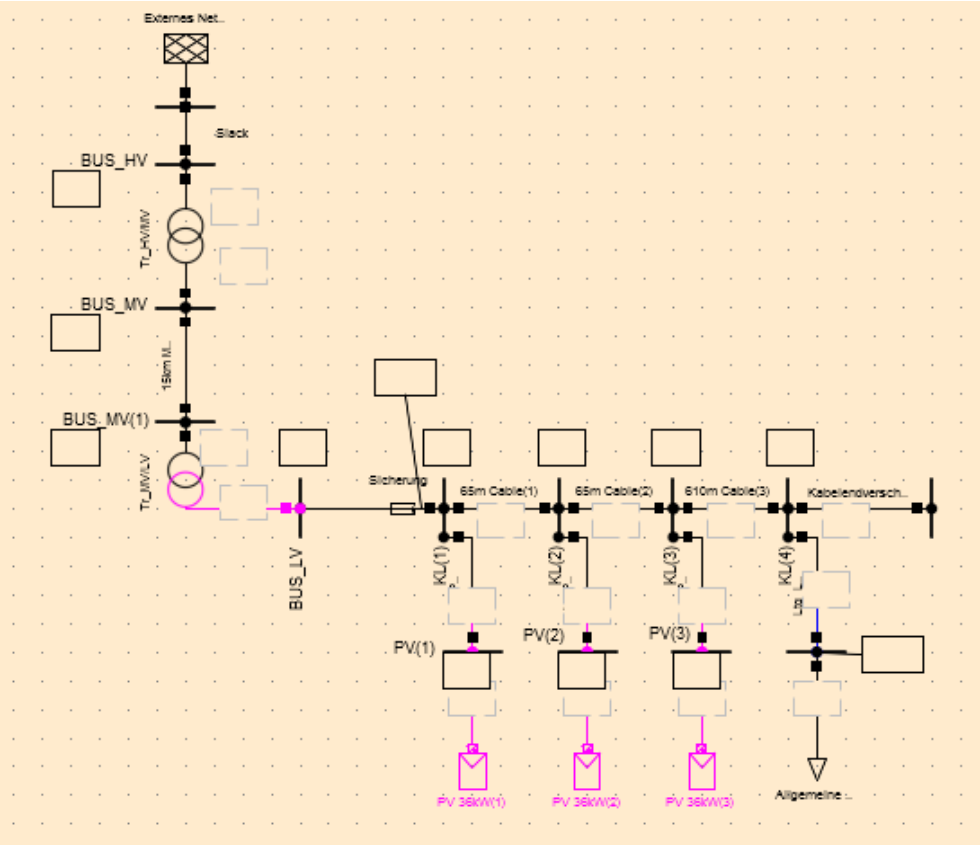

Bild 2: PowerFactory: grid layout ; active external OPC measurement devices are colored in pink

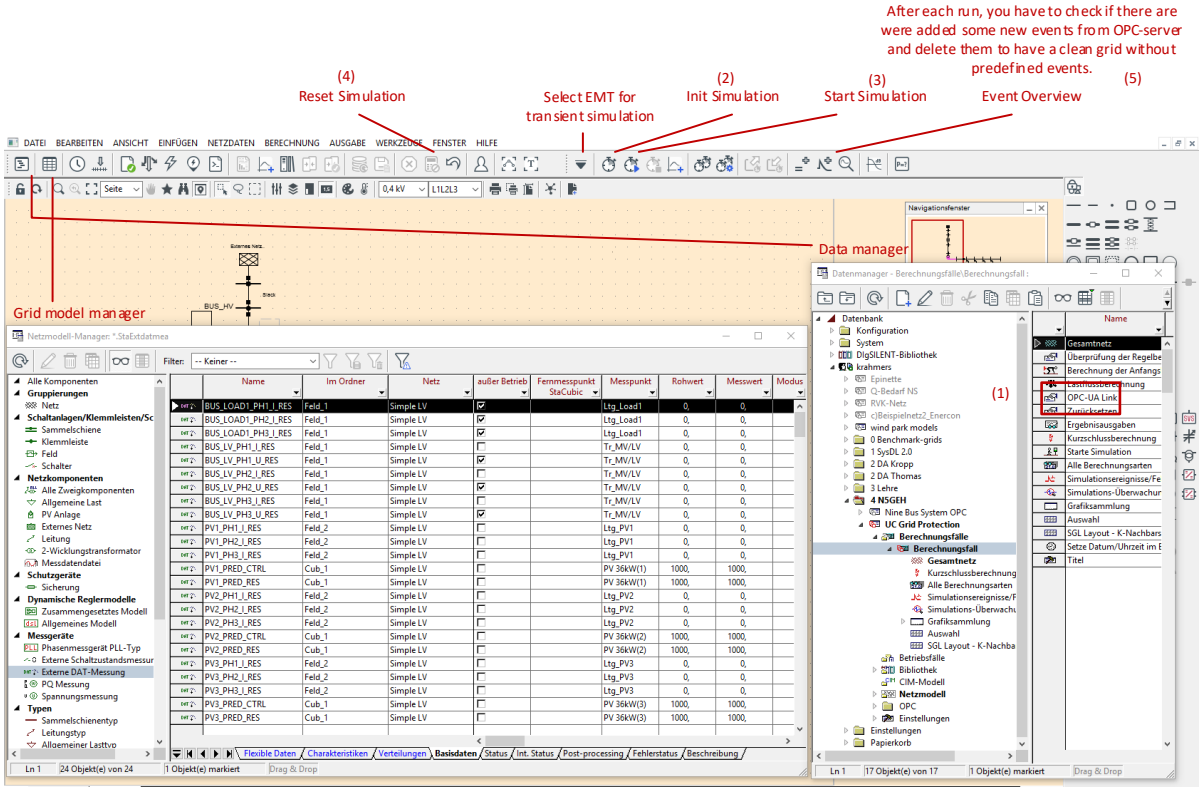

Bild 3: PowerFactory: control panel

Tabelle 1: Overview of external OPC measurement devices; RES – write to server only; CTRL – read from server only

OPC tag	Description	Active
BUS_LV_PH1_I_RES	Slack, Current Phase1	X
BUS_LV_PH2_I_RES	Slack, Current Phase2	X
BUS_LV_PH3_I_RES	Slack, Current Phase3	X
BUS_LV_BREAKER_CTRL	Slack, Circuit Breaker [0-open, 1-closed]	X
BUS_LV_BREAKER_RES	Slack, Circuit Breaker [0-open, 1-closed]	X
PV1_PH1_I_RES	PV1, Current Phase1	X
PV1_PH2_I_RES	PV1, Current Phase2	X
PV1_PH3_I_RES	PV1, Current Phase3	X
PV2_PH1_I_RES	PV2, Current Phase1	X
PV2_PH2_I_RES	PV2, Current Phase2	X
PV2_PH3_I_RES	PV2, Current Phase3	X
PV3_PH1_I_RES	PV3, Current Phase1	X
PV3_PH2_I_RES	PV3, Current Phase2	X
PV3_PH3_I_RES	PV3, Current Phase3	X
PV1_PRED_CTRL	PV1, Active Power Reduction via SolarRadiation in 0,1%	X
PV1_PRED_RES	PV1, Active Power Reduction via SolarRadiation in 0,1%	X
PV2_PRED_CTRL	PV2, Active Power Reduction via SolarRadiation in 0,1%	X
PV2_PRED_RES	PV2, Active Power Reduction via SolarRadiation in 0,1%	X
PV3_PRED_CTRL	PV3, Active Power Reduction via SolarRadiation in 0,1%	X
PV3_PRED_RES	PV3, Active Power Reduction via SolarRadiation in 0,1%	X
BUS_LOAD1_PH1_I_RES	LOAD1, Current Phase1	
BUS_LOAD1_PH2_I_RES	LOAD1, Current Phase2	
BUS_LOAD1_PH3_I_RES	LOAD1, Current Phase2	
BUS_LV_PH1_U_RES	Slack, Voltage Phase1	
BUS_LV_PH2_U_RES	Slack, Voltage Phase2	
BUS_LV_PH3_U_RES	Slack, Voltage Phase3	

run main.py

- starts a custom OPC-Server
  - this will create a new OPC node for each tag available in PowerFactory based on PF_ExtMeas_GridProtection.txt
  - make them writable
  - from now one can observe OPC nodes at server by using the external uaExpert-Client
- start DataHandler and pass TopologyFile (path)
  - map nodes of TopologyFile with all reachable nodes of server
  - starts custom OPC-Clients and make subscription for specific nodes
  - build dataframe based on incoming data
  - check if dataframe contains at least one valid dataset for all nodes
    - starts DiffCore by passing dataframe and handle of OPC-Client

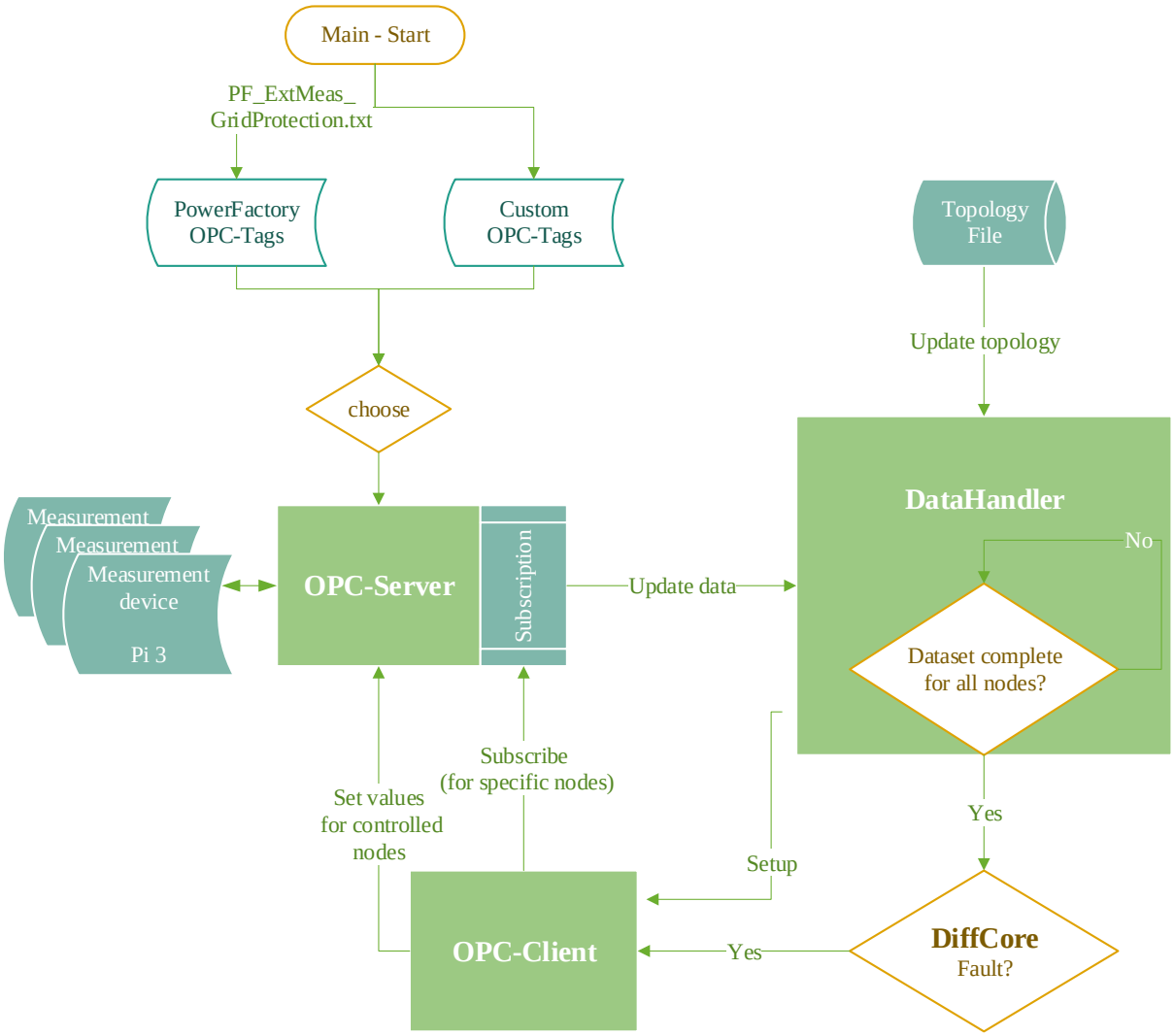

Bild 4: Flow chart for cloud setup; OPC-Client could be replaced by kind of context broker