ViPER/OPC Server Integration

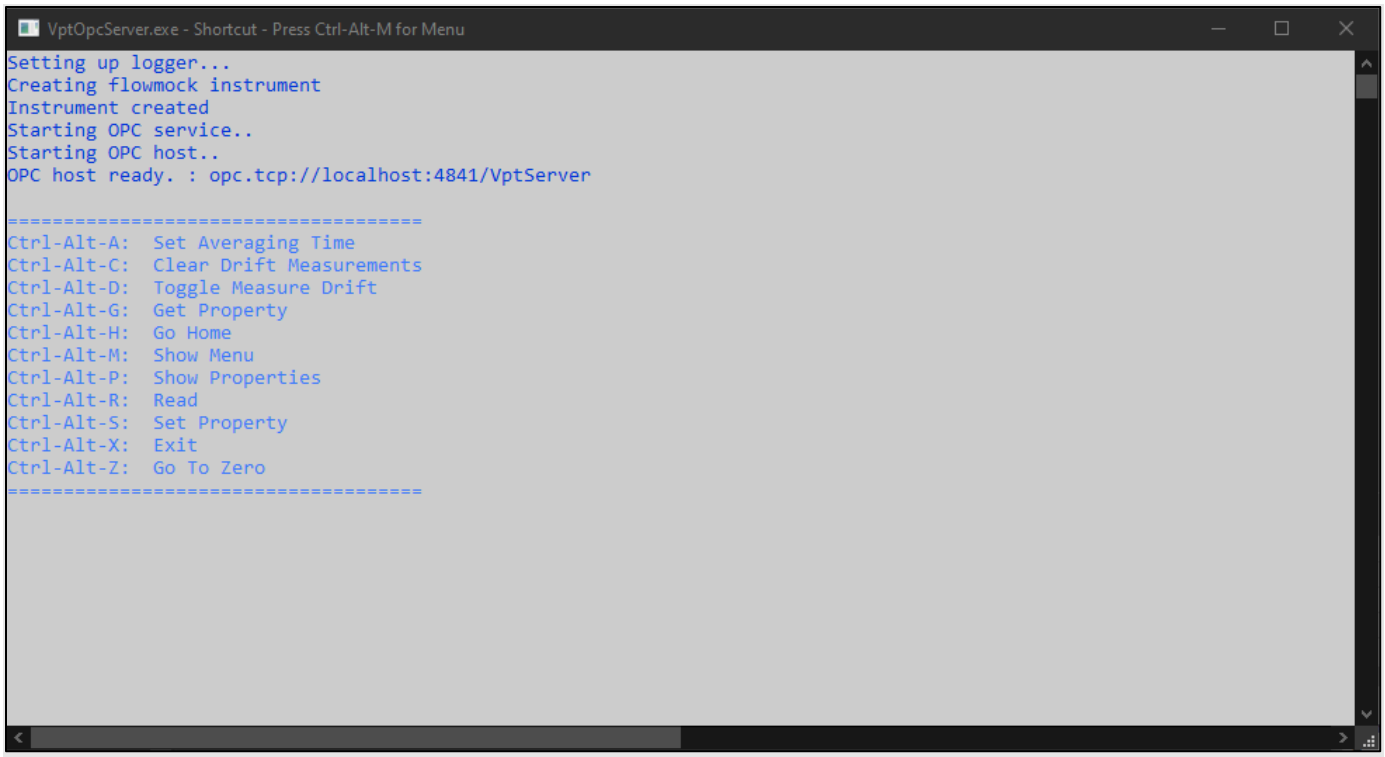
Abstract:	The purpose of this document is to cite the features of ViPER™ VPTCore service that controls the instrument functionality through a generic OPC client which can be used as a basis to implement with other OPC clients.
Applicability:	This procedure applies to developers that are tasked with interfacing VPTCore functionality with an OPC client.
Symptom:	N/A
Cause:	N/A

Definitions:

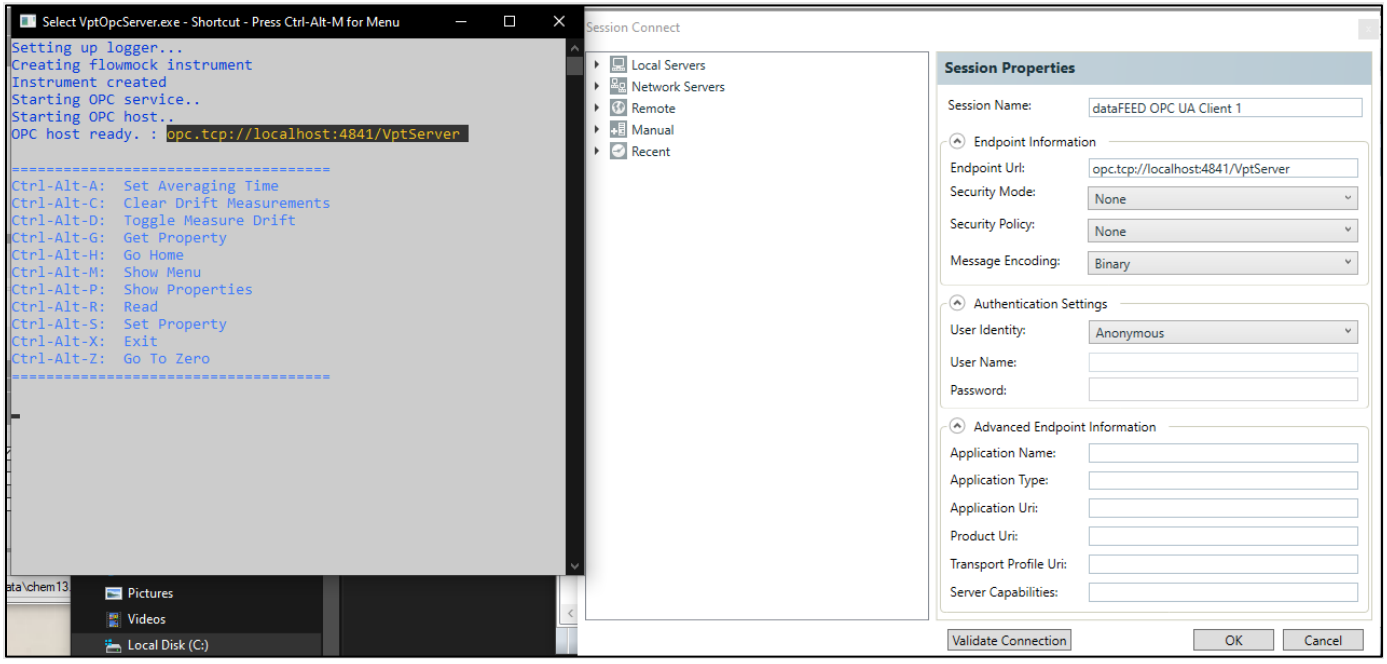
- **OPC UA:** Open Platform Communications Unified Architecture
- **ViPER Software:** Variable Pathlength Ecosystem Release; a bundled software release consisting of the core ViPER Software and the Secure Add-On
    - **VPTCore:** Part of the ViPER Software; handles the communication between user interfaces and the hardware logic
- **JSON:** JavaScript Object Notation;a commonly used format when exchanging data between layers
- **Methods:** Used by the software as instructions or parameters for the instrument to follow to take a slope measurement
- **Quick Method:** Uses the Quick Slope algorithm to find the best slope/concentration values
- **Fixed Method:** Instructs the instrument to travel to a specified pathlength to start its slope collection
- **DAQ:** Data Acquisition

Initiating Server and Client Communication:

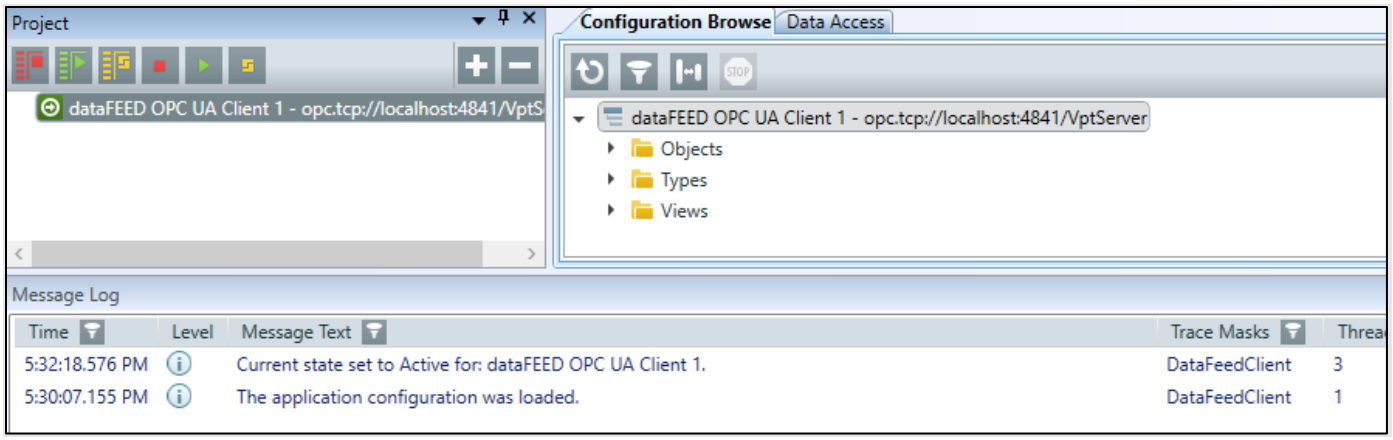
1. Make sure that the Cary 60 spectrophotometer is powered on and the VPT instrument is properly installed prior to starting the software.
2. Once started, the server will populate in a console window with some basic functionalities that can be run directly on the server for troubleshooting purposes.



3. Start the client. In this case, the generic client, dataFEED, is used.
4. Under the Endpoint URL, enter the address that the OPC server has started, to connect the client to the server. In the example below, the address is opc.tcp://localhost:4841/VptServer

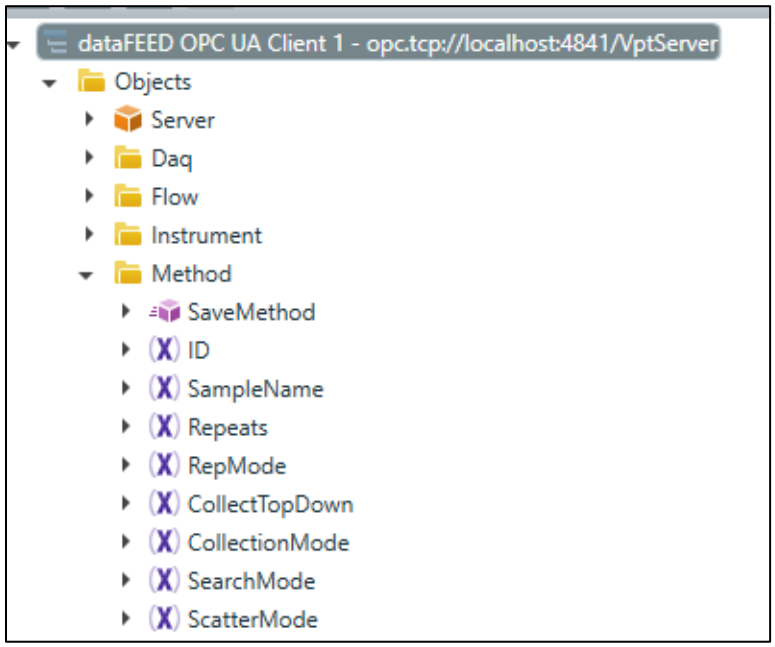


5. Once the session is connected, the node structure is revealed in a new window, and access is available to the server and all of its nodes.



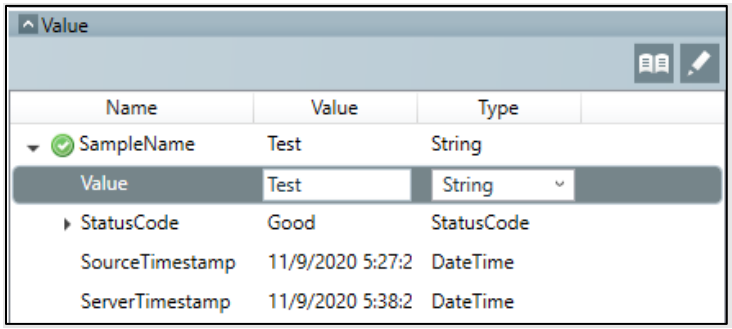
Creating a Method:

1. Under the Objects node, open the Method node. It reveals a list of options for the Method, as shown in the screenshot below.

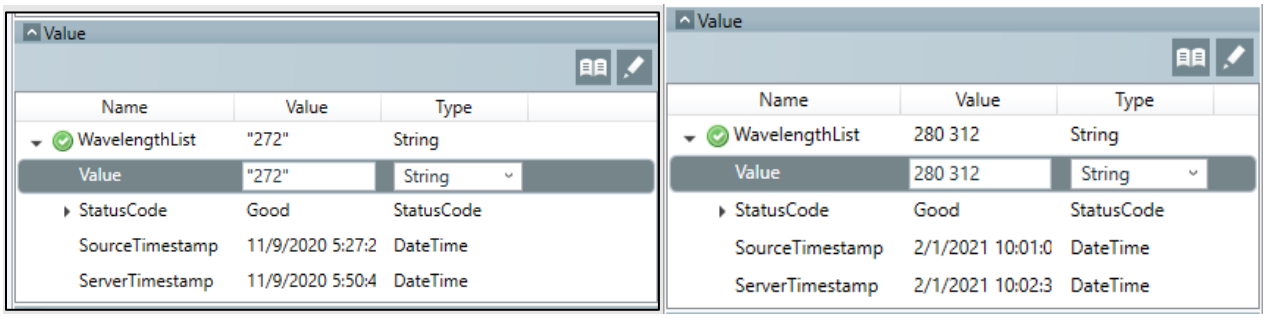


List of the properties that Quick and Fixed Methods share: SampleName, RepMode, Repeats, WavelengthList, ExtinctionCoeffcientList, ScatterMode, MultiDatapointsList, and SearchMode.

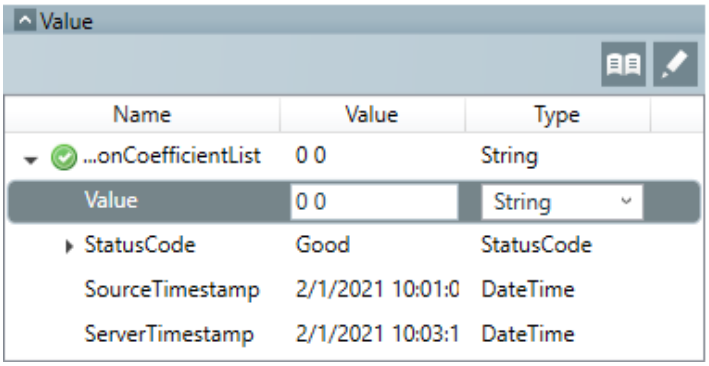
2. **SampleName** property identifies the run. Users make this entry unique for their day-to-day run at their discretion, but in the database, the field is not unique. This property accepts a string input.



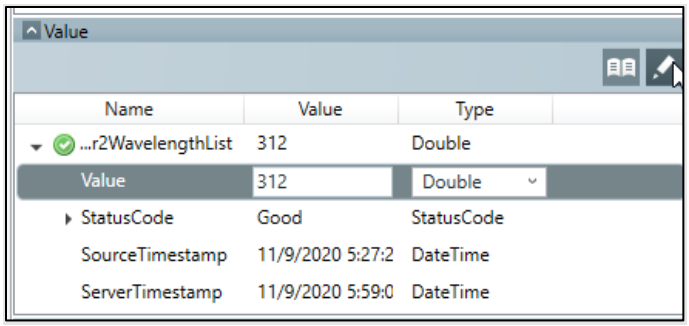
3. **RepMode** defines how the system performs repetitions of a method. The possible integer inputs for the different modes of method repetition are
    - 0 = Off, will run the method only once,
    - 1 = Repeat, will run the method to a specified amount (specified by the Repeats property),
    - 2 = Replicates, similar to Repeat but pauses between runs for the user to change the Fibrette and/or sample before continuing (**Note:** Not applicable to the FlowVPX™ System), and
    - 4 = Continuous, runs the method continuously until the method is manually stopped.
4. **Repeats** defines the amount of times the method will run. If RepMode is set to 0 or 4 (Off or Continuous), this field should be set to 0. This property accepts an integer input.
5. **WavelengthList** defines the wavelengths at which the spectrophotometer takes its readings. If a single wavelength is defined, wrap the input (which is usually a double-type) in quotes. But if multiple wavelengths are being defined, separate each wavelength with a space ( ). This property takes a string as input.



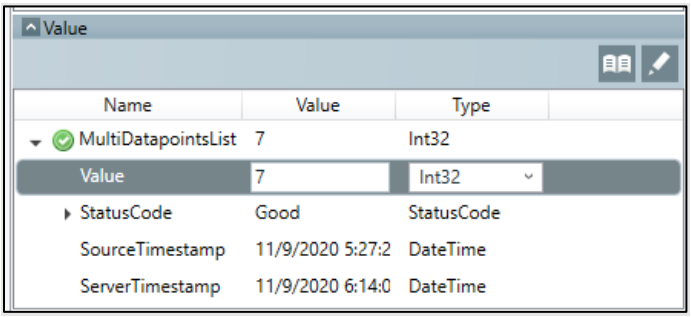
6. **ExtinctionCoefficientList** defines the extinction coefficient of the sample per wavelength. This value allows the software to calculate the concentration for every slope. Therefore, if there is n amount of wavelengths, there should also be n amount of extinction coefficients. Similar to the WavelengthList value, the extinction coefficient value must be wrapped in quotes if using one value or separated with a space for multiple values. If the extinction coefficient is unknown, 0 can be entered as a placeholder value. This field accepts a string as input.



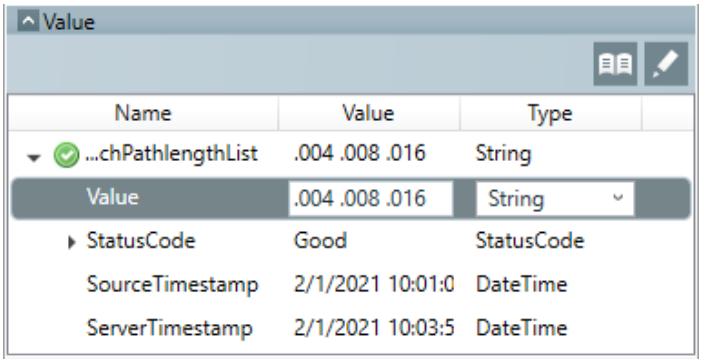
7. **ScatterMode** defines the type of scatter correction (single wavelength or dual wavelength) that will be applied for each slope collection. The acceptable values are 0 = Off, 1 = Single, 2 = Dual. This property accepts an integer as input.
    - If ScatterMode is set to 1 (Single), a value for **Scatter1WavelengthList** is required. If ScatterMode is set to 2 (Dual), values for **Scatter1WavelengthList** and **Scatter2WavelengthList** are both required. Both accept a double-type as input.



8. **MultiDatapointsList** defines how many readings the Cary 60 spectrophotometer measures to complete one slope. The best practice is to set a lower limit of five data points, but technically only two data points are needed to create a slope between them. This property accepts an integer as input.



9. **SearchMode** defines the type of method that will run: 0 = Quick, or 1 = Fixed. This property accepts an integer as input.
10. Quick Slope specific properties:
    - When SearchMode is set to 0, a value for **TargetThresholdAbsorbance** must then be defined. This property defines the absorbance value that the Quick Slope algorithm aims for when searching for the most optimal absorbance to start the slope collection. This property accepts a double-type as input.
    - **SearchPathlengthList** defines the pathlengths at which it attempts to find the specified TargetThresholdAbsorbance. Normally, three pathlength values (double-type) are input to have the best outcome. This property accepts a space-separated string as input.

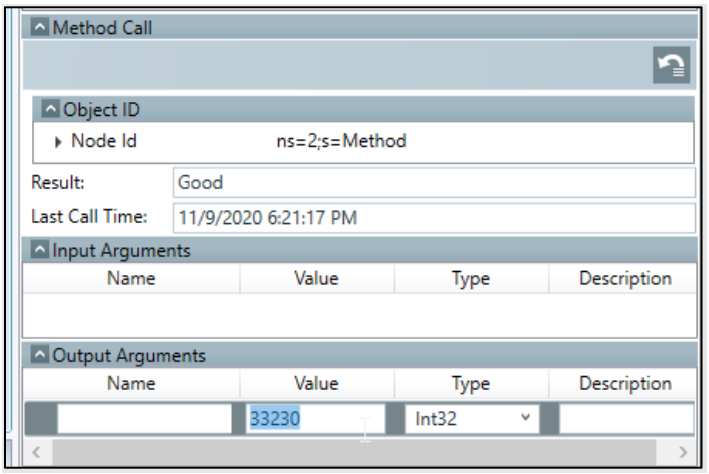


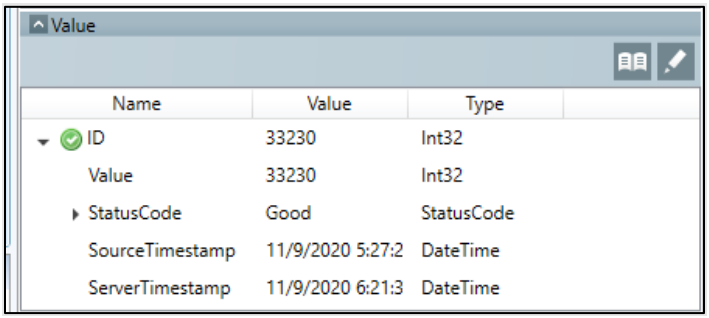
11. Fixed Slope specific properties:
    - When SearchMode is set to 1, **StartingPathlengthsList** must then be defined. This informs the VPT instrument at which pathlength to start the slope collection. This property accepts a double-type as input.
    - **StepPathlengthList** defines the steps from one data point to another. For example, if the value for StartingPathlengthsList is specified to 1.000 mm and the value for StepPathlengthList is specified to 0.001 mm with five data points, the VPT instrument first takes a measurement at 1.000 mm, steps

down to 0.999 mm and performs a reading, then to 0.998 mm and performs a reading, and so on until five data points have been collected.

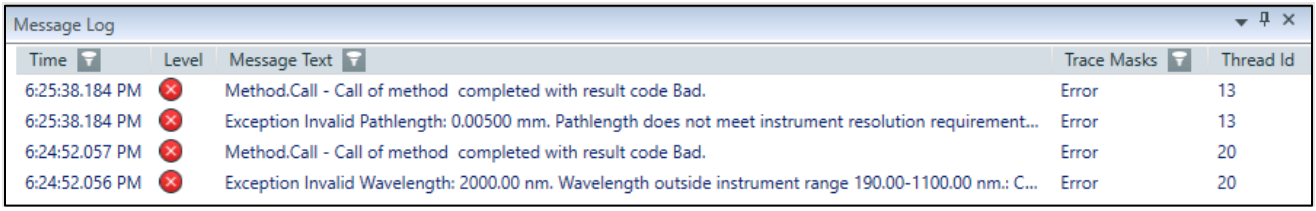
Saving a Method:

1. Once the necessary properties are filled in, the method can be saved using the SaveMethod feature. To call the feature, press the arrow button under Method Call. Once saved, the method ID will be part of the output, as highlighted in the screenshot below under Value, and the ID node will be automatically populated with the method ID.



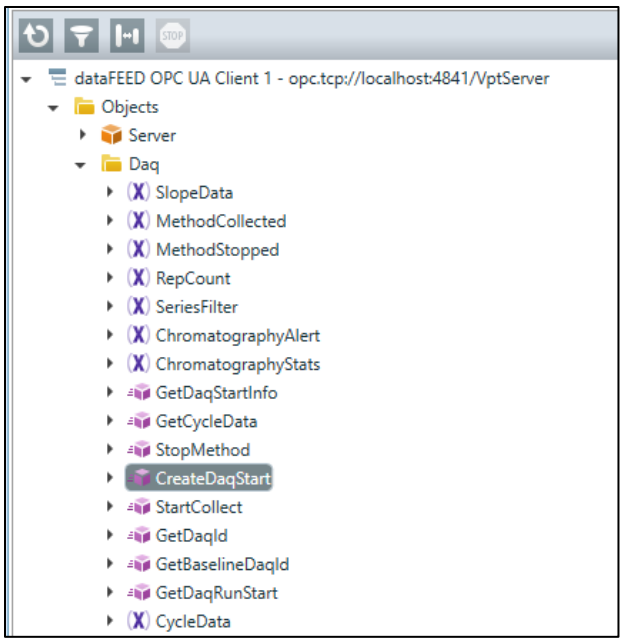


2. The Message Log displays any errors that may have been identified by the software. Reviewing these entries may give guidance on taking corrective actions. In the case below, the WavelengthList was out of acceptable range.

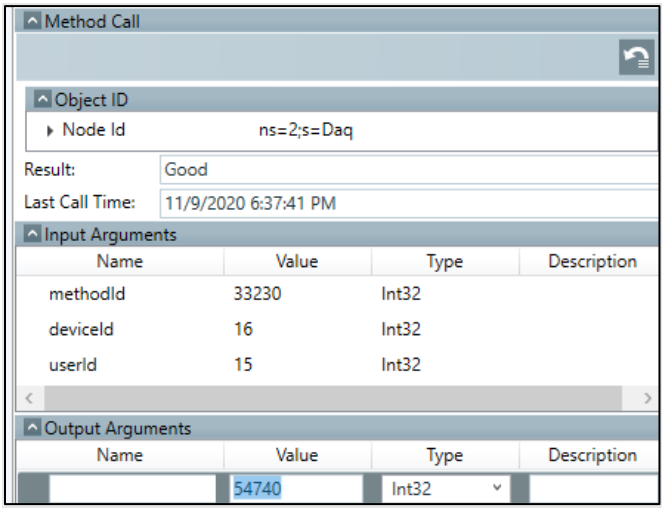


Running a Method:

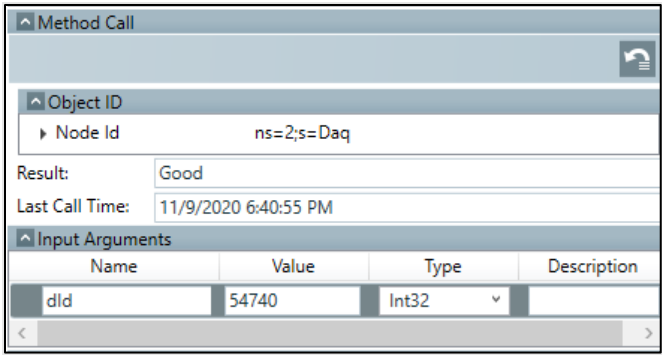
1. Make sure that the VPT instrument is loaded before starting a run.
2. To start the method, first create an entry (called the DAQ entry). Navigate to Objects > Daq, and then select CreateDaqStart. Starting a method requires an input of the method ID number (methodId), the ID number of the device running the method (deviceId), and the ID number of the user running the method (userId). All of the ID properties accept integers as inputs.



3. Once called, it will output an ID number which signifies the ID of the DAQ (DaqId), as highlighted in the screenshot below.



4. To begin running the method, copy the DaqId output value (highlighted above), open the StartCollect node, input the DAQ ID number, and call the StartCollect using the button under Method Call. This will run the method that was tied to that DAQ.

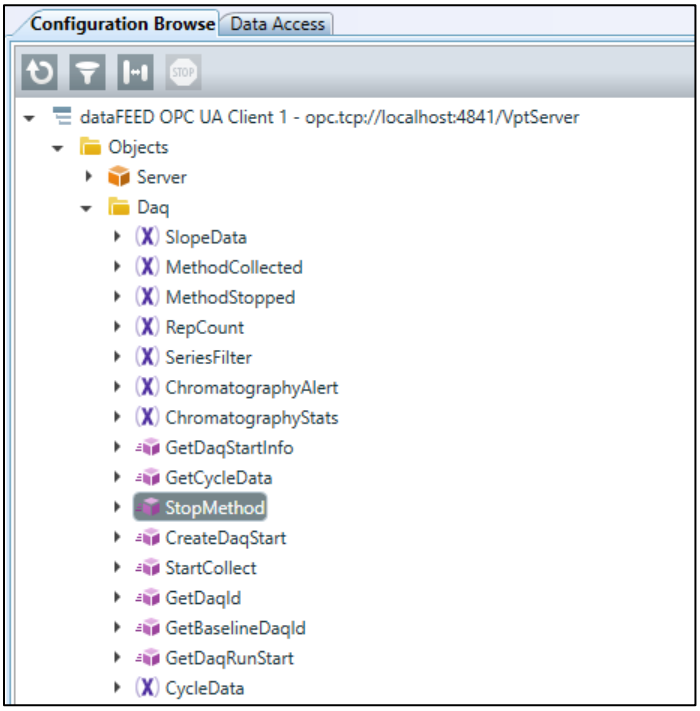


5. The VPT instrument and Cary 60 will begin collecting data, and the server console outputs the data as it is collected.

```
PhotoData: 280.000000000000 0.599291384000 0.000000000000 thx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.054460198000 0.000000000000 thx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.599291384000 0.000000000000 thx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.101763994000 0.000000000000 thx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.223282456000 0.000000000000 thx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.367449075000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.415677398000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.461633325000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.485983998000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.485983998000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.485983998000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
PhotoData: 280.000000000000 0.485983998000 0.000000000000 plx{Sample_20201109234355}@280.00nm   Absorbance
The given array is too small. It must be at least 5 long.
Parameter name: x
{
  "SeriesData": [
    {
      "ID": 11754929,
      "DAQID": 54741,
      "CycleCountId": 1,
      "RawDataId": null,
      "Wavelength": 280.0,
      "Pathlength": 0.752000000000000000000000000,
      "Absorbance": 0.367449075,
      "PercentT": null,
      "Temperature": 0.0,
      "Ph": null,
      "Humidity": null,
```


Stopping a Method:

1. If the RepMode value is set to 0, 1, or 2, the method will stop automatically after a set number of repeated runs. If the RepMode value is set to 4 (Continuous), then it is necessary to stop it manually. This can be done by navigating to Objects > Daq > StopMethod, as illustrated in the screenshot below, and calling the node.

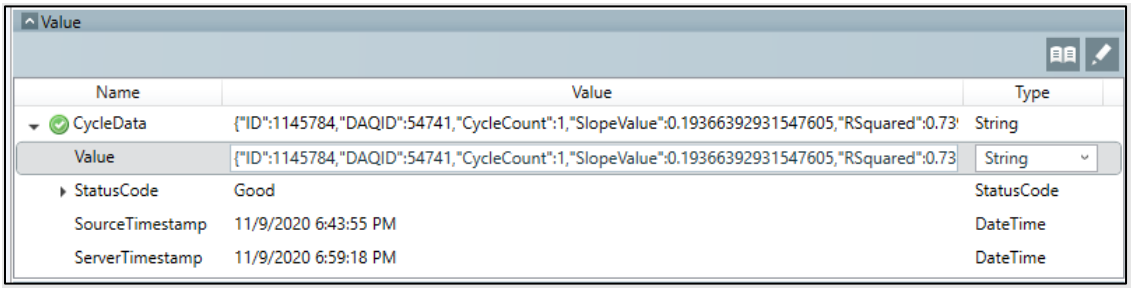


Data Subscription:

The user may also subscribe to the data as it is collected. For compatibility with ViPER's Web-based structure, the data is in JSON format and it may be necessary to parse the data to obtain the relevant data needed. Calling the properties will extract the data specified below.

- **SlopeData:** Subscribing to the /Daq/SlopeData node reports including, but not limited to, the slope, concentration, and R^2 value.
- **CycleData:** Subscribing to the /Daq/CycleData node reports not only the slope data, but also the raw data (the pathlength, wavelength, and the acquired absorbance at each data point) that was used to calculate the slope, as well as when the cycle was run and the user who ran the cycle.

  Example:

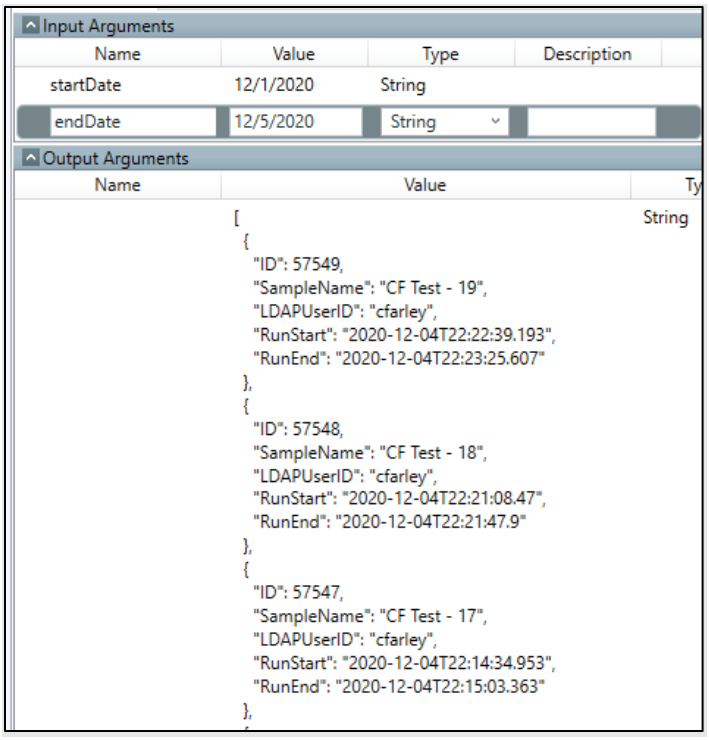


- **MethodCollected:** Subscribing to the /Daq/MethodCollected node notifies the user once a method has been automatically stopped (if RepMode is set to 0, 1, or 2).
- **MethodStopped:** Subscribing to the /Daq/MethodStopped node notifies the user if a method is manually stopped.
- It is also possible to subscribe to certain properties of the device to know the current status of certain properties. For example, subscribing to the /Instrument/IsCollecting node returns the value which indicates if the instrument is in the middle of a method collection.

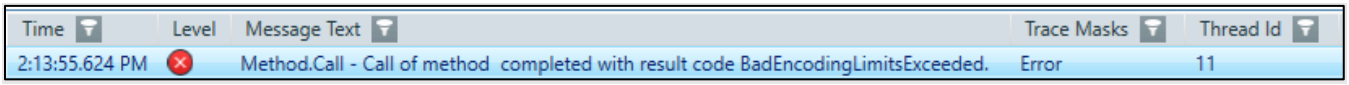
This information can then be used with a Historian so that a user can access previously saved methods and runs.

Historical Data:

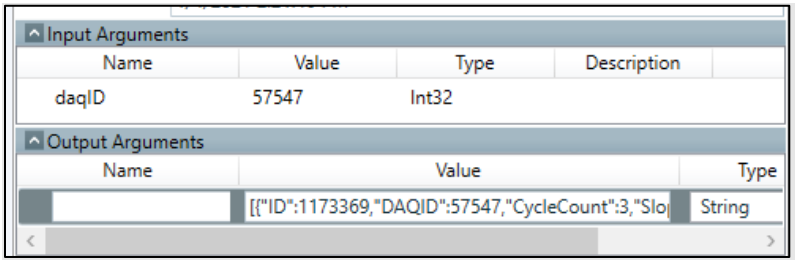
1. It is highly recommended that a Historian be used in conjunction with the data subscription. This allows the data to be easily linked with other data acquired by other systems using the same Historian.
   - If collected data needs to be retrieved at a later time, ViPER's OPC server allows the user to review previously run data. Under the Daq node, select the GetDaqStartInfo node. A start and end date (MM/DD/YYYY format) value can be input, and all runs that have been performed within the specified range will be subsequently retrieved, as shown in the screenshot below.



2. The data is in a JSON array format and shows the most relevant information to limit the size of the string, while still conforming with the format of the OPC UA. Retrieving data from larger date ranges may yield an error, as pictured below, that can be corrected by setting a smaller date range.



3. Once the desired DAQ entry is found, copy the DAQ ID number to retrieve the Cycle/Slope information of that DAQ. In the Daq node, navigate to the GetCycleData node and enter the daqID. This will display all the cycles/slopes acquired for that DAQ.



Document Info: KB20006

Revision History			
Rev	**Date**	**Changes**	**Initials**
00	2020-01-04	Initial Draft	PL

Prepared By:

C Technologies, Inc.

685 Route 202/206 Bridgewater, NJ 08807

+1 908-707-1009

analytics-support@repligen.com