

# **R-Series OPC Python API Reference**

Release 2.1

## **R-SERIES OPC UA CLIENT**

| 1 | Intro | oduction           | 1                       |
|---|-------|--------------------|-------------------------|
|   | 1.1   | Requirements       |                         |
| 2 | R-So  | ries OPC UA Client | 3                       |
| _ | 2.1   |                    |                         |
|   | _,,   |                    |                         |
|   |       |                    |                         |
|   |       | 1                  |                         |
|   |       |                    | es package              |
|   |       | 2.1.2.1.1.1        | autosampler             |
|   |       | 2.1.2.1.1.2        | azura                   |
|   |       | 2.1.2.1.1.3        | chiller                 |
|   |       | 2.1.2.1.1.4        | detector                |
|   |       | 2.1.2.1.1.5        | factories               |
|   |       | 2.1.2.1.1.6        | flowChemistry           |
|   |       | 2.1.2.1.1.7        | ion                     |
|   |       | 2.1.2.1.1.8        | manualControl           |
|   |       | 2.1.2.1.1.9        | mfc                     |
|   |       | 2.1.2.1.1.10       | module                  |
|   |       | 2.1.2.1.1.11       | pump                    |
|   |       | 2.1.2.1.1.12       | r2                      |
|   |       | 2.1.2.1.1.13       | r4                      |
|   |       | 2.1.2.1.1.14       | reactor                 |
|   |       | 2.1.2.1.1.15       | sf10                    |
|   |       | 2.1.2.1.1.16       | syringe                 |
|   |       | 2.1.2.1.1.17       | uv150                   |
|   |       | 2.1.2.1.1.18       | vbfr                    |
|   |       | 2.1.2.1.2 experi   | ment package            |
|   |       | 2.1.2.1.2.1        | experiment              |
|   |       | 2.1.2.1.2.2        | experimentSetup         |
|   |       | 2.1.2.1.2.3        | experimentStatistics    |
|   |       | 2.1.2.1.2.4        | reactionManager         |
|   |       | 2.1.2.1.3 identif  | fication package        |
|   |       | 2.1.2.1.3.1        | identification          |
|   |       | 2.1.2.1.3.2        | equipmentIdentification |
|   |       | 2.1.2.1.3.3        | softwareIdentification  |
|   |       |                    | oring package           |
|   |       | 2.1.2.1.4.1        | monitoring              |
|   |       |                    | ation package           |
|   |       | 2.1.2.1.5.1        | notification            |

| Index    |          |       |           |           |       |      |  |  |  |  |  |  |   |  |       |      | 63 |
|----------|----------|-------|-----------|-----------|-------|------|--|--|--|--|--|--|---|--|-------|------|----|
| Python N | Module 1 | Index |           |           |       |      |  |  |  |  |  |  |   |  |       |      | 61 |
|          | 2.1.3    | utils |           |           | <br>• | <br> |  |  |  |  |  |  | • |  |       |      | 59 |
|          |          |       | 2.1.2.1.8 | reaction  | <br>  | <br> |  |  |  |  |  |  |   |  |       | <br> | 58 |
|          |          |       | 2.1.2.1.7 | factories | <br>  | <br> |  |  |  |  |  |  |   |  |       |      | 58 |
|          |          |       | 2.1.2.1.6 | base      |       | <br> |  |  |  |  |  |  |   |  | <br>• | <br> | 58 |

**CHAPTER** 

**ONE** 

## INTRODUCTION

The R-Series OPC UA Python Client is an API to connect the Vapourtec R-Series flow chemistry controller into an OPC UA network. This allows the user to write and run Python scripts.

The Python script could be a way to automate the process and control the R-Series flow chemistry controller in a remote way through the OPC UA network.

The R-Series OPC UA Python client is based on the freeopcua Python package. For further information about this package, see https://python-opcua.readthedocs.io/en/latest/.

## 1.1 Requirements

To run and install the R-Series OPC UA Python Client, the system must have installed a Python 3 version.

Because the Python's popularity, there are a lot of environments for python script developing and execution. For this explanation, we use the Anaconda environment under Windows OS, and the Spyder IDE with is provided in installing Anaconda.

You can feel free to use your favourite Python IDE and environment manager.

For downloading and installing Anaconda, check its website: www.anaconda.com

## **R-SERIES OPC UA CLIENT**

## 2.1 rseriesopc

## 2.1.1 client

Created on Wed Jun 23 14:18:43 2021

This module is for the high level opc client configuration. It contains the connection and disconnection methods to the server.

## class RSeriesClient(url)

The RSeries Client class contructs a OPC client object for RSeries OPC server.

This class inherits opcua. Client class. This allows to use all of the funtions provided by it. Further information about this class and its use cases could be found in https://python-opcua.readthedocs.io/en/latest/

This is a High Level Class that instantiates all the needed components to connect the R-Series Controller through OPC-UA communication

## **Parameters**

**url** (String) – It is the RSeries Controller OPC Server address.

#### rSeries

It represents the R-Series machine. This has all the methods to communicate to the controller.

## **Type**

FlowChemistryType

## isConnected

It is true when the client is connected to the R-Series Controller. Otherwise, this is false.

## Type

bool

## connect()

It connects the client with RSeries OPC Server.

## disconnect()

It disconnects the client with the R-Series Controller OPC UA Server.

## getRSeries()

It returns the RSeries machine

#### connect()

It connects the client with RSeries OPC Server.

#### Returns

It returns true if the connections was successful. Else, it returns false.

## **Return type**

bool

#### disconnect()

It disconnects the client with the R-Series Controller OPC UA Server.

## Return type

None.

#### getRSeries()

It returns the RSeries machine

#### Returns

It is the R-Series machine.

## Return type

*FlowChemistryType* 

## loadRSeries()

This function look up in the server and construct the RSeries object of interest.

#### Returns

It is True when an RSeries machine was loaded. Else, it is False.

## Return type

bool

## 2.1.2 model package

## 2.1.2.1 model

## 2.1.2.1.1 devices package

## 2.1.2.1.1.1 autosampler

## class AutoSamplerType(root)

The AutoSamplerType class instantiates Autosampler objects. This objects allows a high level communication to send orders to the GX271 system, connected to the R-Series Controller.

#### **Parameters**

root (opcua.Node) – It is the parent node in the OPC UA Address Space.

#### root

This contains the id of the Autosampler in the OPC UA Address Space.

## **Type**

opcua.Node

#### collectionValveState

It is the id of the collection valve variable in the OPC UA Address Space.

## **Type**

opcua.Node

## cleanLoop(channel)

It cleans the channel loop

#### **Parameters**

**channel** (*integer*) – This is the channel that will be cleaned.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

## cleanNeedle()

It rinses the needle.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

## cleansSampleLoops()

The probe will inject the solvent from the reservoir and flush the sample loops with the solvent. Depend on the setup, it will activate the number of sample loops as per the experiment setup in R-series controller.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## getCollectionValveState()

It gives if the valve is in collection or waste direction.

#### **Returns**

This is the valve position. If it is true, the valve is in collect position. Else, the valve is in waste position.

## Return type

Bool

#### getCollectionValveStateNode()

This method returns the id of the colection valve variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

This is the id of the collection valve variable in the OPC UA Address Space.

## Return type

opcua.Node

#### goHome()

It moves the Autosampler arm to home position.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## goToPortA()

It moves the autosampler arm to port A.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

## goToVialForCollection(site, time)

It moves the autosampler arm to the especified and take a sample during a given time.

#### **Parameters**

- **site** (*integer*) It is the position where the autosampler collect from.
- time (float) It is the time to collect in seconds.

#### Returns

It is false when the site is bigger than allowed. Otherwise, this is true.

## Return type

bool

## loadLoopVolumeFromVial(channel, position, volume)

It moves the arm to a given position and dispenses product.

#### **Parameters**

- **channel** (*integer*) This is the rack index.
- **position** (*integer*) It is the position in the rack where the product will be dispensed.
- **volume** (*integer*) It is the amount of liquid in ml to dispense into the loop.

#### Returns

It is false when the position is bigger than allowed. Otherwise, it is true.

## **Return type**

bool

## lowerNeedleIntoPortA()

It moves the needle down into Port A.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## **Return type**

bool

## lowerNeedleOverPortA()

It moves the needle down over Port A.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

bool

## lowerSyringePump()

Use this function when you need to replace the syringe. Refer to section 4.11 Syringe pump installation.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

## primeSampleLoop()

It fills the syringe tube with solvent. The Syringe must be primed before running the first reaction and when the solvent needs to be changed.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## riseSyringePump()

Use this function when you need to install the syringe. Refer to section 4.11 Syringe pump installation.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## setCollectionValveState(state)

It sets the autosampler valve in collection or waste position.

#### **Parameters**

**state** (*Bool*) – This is the postion of the valve. When it is true, the valve will be set on collect position. Otherwise it will be set on waste position.

## Return type

None.

## stopRoutines()

This method stops any routine execution.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## switchValveToCollect()

It switches the Waste/Collect to Collect and a few seconds later, the valve comes back to waste in an autmatic way.

## Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## 2.1.2.1.1.2 azura

## class AzuraPumpType(root)

An AzuraPumpType object contains all of the methods and object to monitor and control a Azura Pump Module in the R-Series Controller.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the R2 object id in the OPC UA Address Space.

## **Type**

opcua.Node

#### erroState

It is the ErrorState variable id in the OPC UA Address Space.

#### **Type**

opcua.Node

#### maxPressureLimit

It is the MaxPressureLimit variable id in the OPC UA Address Space.

#### **Type**

opcua.Node

## pressureLimit

It is the PressureLimit variable id in the OPC UA Address Space.

#### **Type**

opcua.Node

## flowRate

It is the FlowRate variable id in te OPC UA Address Space.

## Type

opcua.Node

## getErrorState()

It reports if there was an error with the module.

## Returns

It is True if there was an error. Otherwise, it si False.

#### Return type

bool

## getErrorStateNode()

This method returns the id of the ErrorState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the ErrorState variable in the OPC UA Address Space.

opcua.Node

## getFlowRate()

This function gives the current pump flow rate.

#### Returns

It is the pump flow rate in ul/min.

#### Return type

integer

## getFlowRateNode()

This method returns the id of the flow rate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the flowRate variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getMaxPressureLimit()

It gives the actual maximum pressure limit.

## Returns

It is the actual maximum pressure limit.

## Return type

float

#### getMaxPressureLimitNode()

This method returns the id of the SF10 MaxPressureLimit variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 MaxPressureLimit variable in the OPC UA Address Space.

## **Return type**

opcua.Node

## setFlowRate(flowRate)

It sets the flow rate to the pump. When flow rate is setted in 0, the pump is turned off. If the pump is setted in other value than 0, the pump is immediatly turned on. The flow rate must be set in a suitable value for the connected pump in R2. In other case, the flow rate will not be setted in the machine.

#### **Parameters**

**flowRate** (integer) – It is the pump flow rate in ul/min.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## **Return type**

bool

## setMaxPressureLimit(newMaxPressureLimit)

It sets a new maximum pressure limit.

#### **Parameters**

**newMaxPressureLimit** (*float*) – It is the new maximum pressure limit.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

#### 2.1.2.1.1.3 chiller

## class ChillerType(root)

A ChillerType Object contains all the variables and methods to know the measurements and information about the Chillers.

#### **Parameters**

root (opcua. Node) – It is the id of a Chiller object in the OPC UA Address Space.

#### root

It is the id of the Chiller object in the OPC UA Address Space.

## **Type**

opcua.Node

#### errorState

It is the id of the error state variable for the Chiller in the OPC UA Address Space.

## **Type**

opcua.Node

## ${\tt setTemperature}$

It is the id of the Chiller Temperature Set Point variable in the OPC UA Address Space.

#### **Type**

opcua.Node

## temperature

It is the id of the Chiller Temperature measurement variable in the OPC UA Address Space.

## **Type**

opcua.Node

#### chillerType

It is the id of the type of Chiller variable in the OPC UA Address Space.

#### **Type**

opcua.Node

## leaveRunning

It is the id of the Chiller's leaveRunning variable in the OPC UA Address Space.

## Type

opcua.Node

## sensorType

It is the id of the sensor Type of the Chiller variable in the OPC UA Address Space.

## **Type**

opcua.Node

## getChillerType()

It gives the chiller type target.

## Returns

It is the chiller type. 0 means Julabo Chiller. 1 means Huber Chiller.

## Return type

integer

## getChillerTypeNode()

This method returns the id of the Chiller Type variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

This is the id of the Chiller Type variable in the OPC UA Address Space.

## Return type

opcua.Node

## getErrorState()

It reports if there are an error in the chiller.

## Returns

It is True when an error occurs. Otherwise, it is False.

## Return type

bool

#### getErrorStateNode()

This method returns the id of the Chiller's error state variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in reading Data Example.py

## Returns

This is the id of the ErrorState Chiller's variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getLeaveRunning()

It gives if the chiller will be left running after reaction list excecution.

#### Returns

If it is True, the chiller will not be turned off at the end of the reaction lsit excecution.

#### Return type

bool

## getLeaveRunningNode()

This method returns the id of the Chiller's Leave Running variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the LeaveRunning variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getSensorType()

It gives the sensor type present on the device.

#### Returns

It is kind of control for the chiller. 0 means the control is made with the interal sensor. 1 means the control is made with an external sensor.

## Return type

integer

## getSensorTypeNode()

This method returns the id of the Chiller's Sensor Type variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the Sensor Type variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getSetTemperature()

It gives the temperature target.

#### Returns

It is the temperature target for the chiller.

## Return type

float

## getSetTemperatureNode()

This method returns the id of the Chiller's temperature target variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SetTemperature variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getTemperature()

It gives the temperature measurement.

#### Returns

It is the temperature in the chiller.

#### **Return type**

float

## getTemperatureNode()

This method returns the id of the Chiller's temperature variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

This is the id of the Temperature variable in the OPC UA Address Space.

#### Return type

opcua.Node

## setChillerType(chiller)

It set the chiller controller. Each brand needs an specific controller.

#### **Parameters**

**chiller** (integer.) – This is the chiller type. 0 is for Julabo Chiller. 1 is for Huber Chiller.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## setLeaveRunning(state)

It set the leave running parameter.

When Leave Running is True, the chiller will not be turned off at the end of the reaction list excecution.

## **Parameters**

**state** (*bool*.) – This is the new leave running state for the chiller.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

bool

## setSensorType(sensor)

It set the temperature target.

#### **Parameters**

**sensor** (*int*.) – This is the new sensorType for the chiller. 0 means internal sensor. 1 menas external sensor.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## setTemperature(temperature)

It set the temperature target.

#### **Parameters**

**temperature** (*float*.) – This is the new temperature target for the chiller.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## 2.1.2.1.1.4 detector

## class DetectorType(root)

A DetectorType object contains the variables with the measurements and information about the UV50D detector

#### **Parameters**

root (opcua. Node) – It is the id of this object in the OPC UA Address Space.

#### root

It is the id of the detector in the OPC UA Address Space.

## Type

opcua.Node

## lampState

It is the id of the lamp state variable in the OPC UA Address Space.

## **Type**

opcua.Node

## Wavelengths

It is the id of the measuring values in the OPC UA Address Space.

#### **Type**

opcua.Node

## getLampState()

It gives the lamp status. It could be:

OFF = 0

HEATING = 1

ON = 2

#### Returns

It is the lamp status.

## **Return type**

integer

## getLampStateNode()

This method returns the id of the UV50D Lamp status variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

This is the id of the UV50D Lamp Status variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getWavelengthAt(index)

It gives a measuremnt of the desired wavelength detector.

#### **Parameters**

**index** (*integer*) – It is the index of the detector to measure.

#### Returns

It is the detector measurement.

## **Return type**

float

## getWavelengthNode()

This method returns the id of the UV50D measurements variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

This is the id of the UV50D measurements variable in the OPC UA Address Space.

## Return type

opcua.Node

## getWavelengths()

It gives an array with all of detetors measurements.

#### Returns

These are the measurements.

## Return type

list of float

## 2.1.2.1.1.5 factories

## class ModuleTypeFactory

It is a factory that returns the right type of reagent source class when makeReagentSourceType is called. The ModuleType class has any members.

## isAReagentSourceModule()

It reports if the selected node is a module that can be created by this object or not.

#### **Parameters**

**node** (opcua.Node) – It is the node that wants to be readed from OPC UA address space.

#### Returns

It is True when the node is a reagent source object. Otherwise, it is False.

#### **Return type**

bool

## makeReagentSourceType()

It returns an object of one of the reagent source types.

#### **Parameters**

**node** (opcua. Node) – It is the node to be readed from OPC UA address space.

#### **Returns**

It is the reagent source type object

## Return type

R2Type, SF10Type, MFCType, AzuraPumpType, SyringePumpType or None

## 2.1.2.1.1.6 flowChemistry

## class FlowChemistryType(parentNode)

The FlowChemistry class is the representation of the whole Vapourtec R-Series Flow Chemistry machines.

This OPC UA Client is based on the freeopcua. The Python OPC-UA documentation could be found at python-opcua.readthedocs.io.

## Parameters

**parentNode** (*opcua.Node*) – This is the root object of the R-Series OPC UA Server. From there, all of the relative information about the R-Series Flow Chemistry Machine will be read on connection.

#### root

It is the address of the R-Series machine in the OPC-UA Address Space.

#### Type

opcua.Node

## identification

It is an object to get the machine identification information.

#### **Type**

*IdentificationType* 

#### monitoring

It is an object that has the methods to get the data to monitor the R-Series status.

## **Type**

MonitoringType

## experiment

It cointains the experiment settings and the reactions information.

#### Type

ExperimentType

#### notification

This object has address from variables for subscribe to data changes.

#### Туре

*NotificationType* 

#### manualControl

It contains all the methods for low level controlling of the R-Series subsystems.

#### Type

*ManualControlType* 

## getExperiment()

It is a getter to gain easier access to the experiment information.

#### Returns

It is the node that allows the user managing the experiment setup and data.

## **Return type**

ExperimentType

## getIdentification()

It is a getter to gain easier access to the identification information.

## Returns

It is the identification information of the Vapourtec RSeries Flow Chemistry Machine.

## Return type

*IdentificationType* 

#### getManualControl()

It is a getter to gain easier access to manual control methods.

#### Returns

It is the manual control object.

## **Return type**

*ManualControlType* 

## getMonitoring()

It is a getter to gain easier access to the monitoring information.

#### Returns

It is the node that contains the machine status information.

## Return type

*MonitoringType* 

## getNotification()

It is a getter to gain easier access to alerts.

#### Returns

It is the node that emits alerts.

*NotificationType* 

#### 2.1.2.1.1.7 ion

#### class IONType(root: Node)

The IONType class implements the low level functions for controlling the components of the R-Series Machine.

This class allows the user to control the electrical control functions for the Vapourtec ION Electrochemial Reactor.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

This is the parent node in the OPC UA Address Space.

## **Type**

opcua.Node

## voltage

This is the id of the voltage variable in the OPC UA Address Space.

## **Type**

opcua.Node

#### current

It is the id of the current variable in the OPC UA Address Space.

## **Type**

opcua.Node

## getCurrent()

It gives a measuremnt of the ION Reactor's current.

#### Returns

It is the reactor's current measurement.

#### **Return type**

float

## getCurrentNode()

This method returns the id of the ION Reactor's current variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

This is the id of the reactor's current variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getVoltage()

It gives a measuremnt of the ION Reactor's voltage.

#### Returns

It is the reactor's voltage measurement.

## **Return type**

float

## getVoltageNode()

This method returns the id of the ION Reactor's voltage variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the reactor's voltage variable in the OPC UA Address Space.

## Return type

opcua.Node

## setCurrent(newCurrentInA: float)

It sets a new target for the ION Reactor's current.

#### **Parameters**

**index** (*float*) – It is the new current target in Amperes.

#### Returns

It returns a boolean. It it is true, means the value was set. Otherwise, it is false

## Return type

bool

## setVoltage(newVoltageInV)

It sets a new target for the ION Reactor's voltage.

#### **Parameters**

index (float) - It is the new voltage target in Volt.

## Returns

It returns a boolean. It it is true, means the new value was set. Otherwise, it is false

## Return type

bool

#### start()

It starts the electrochemical control functions of the reactor.

## stop()

It stops the electrochemical control functions of the reactor.

#### 2.1.2.1.1.8 manualControl

## class ManualControlType(root: Node)

The ManualControlType class implements the low level functions for controlling the components of the R-Series Machine.

This class allows the user to write and run scripts for custom automation routines.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

This is the parent node in the OPC UA Address Space.

## Type

opcua.Node

#### module

It is a collection of Module Type objects. Those objects contains the reagent source modules and the methods related to control the pumps.

## **Type**

list of a reagent source type

## r4

It is a collection of R4Type objects. Those objects contains the R4 modules and the methods related to set the power and temperatures of the reactors.

#### **Type**

dict of R4Type

#### uv150

This is an object to control UV150 reactors.

#### **Type**

UV150Type

## autoSampler

It is an object that implements the methods to control the AutoSampler module

## **Type**

*AutoSamplerType* 

#### wcValveState

It is an the WCValveState variable in the OPC UA address space. It can control the Waste/Collect valve.

## Type

opcua.Node

#### chiller

It is a collection of ChillerType bbjects. Those objects contains the chiller modules and the methods related to control and monitoring them.

## Type

dict of ChillerType

## getAutoSampler()

If there is an AutoSampler module, this gives that node.

## Returns

If the AutoSmapler is in RSeries, it is returned.

AutoSamplerType or None

## getChiller1()

If there is a Chiller, this gives that node.

#### Returns

If Chiller is in RSeries, it is returned. Else, None is returned

## Return type

ChillerType or None

## getChiller2()

If there is a Chiller, this gives that node.

#### Returns

If Chiller is in RSeries, it is returned. Else, None is returned

## Return type

ChillerType or None

## getChillers()

It returns a dictionary with all Chillers modules.

#### Returns

These are all of the Chillers modules.

## Return type

dict

## getDetector()

It returns an external detector object.

## Returns

It is the detector.

## Return type

DetectorType

## getR4()

It returns a dictionary with all R4 modules.

## Returns

These are all of the R4 modules.

## **Return type**

dict

## getR4I()

If there is a R4 Module, this gives that node.

#### Returns

If R4Module is in RSeries, it is returned. Else, None is returned

## Return type

R4Type or None

## getR4II()

If there is a second R4 Module, this gives that node.

#### Returns

If a second R4Module is in RSeries, it is returned. Else, None is returned

R4Type or None

## getUV150()

If there is a UV150 module, this gives that node.

#### Returns

If a UV150module is in RSeries, it is returned.

#### Return type

UV150Type or None

## getVBFR()

It gives the Variable Bed Flow Reactor object.

## Returns

It is the VBFR.

## Return type

**VBFRType** 

## getWCValveState()

This gives the state of the valve

#### Returns

True means liquid from Collect to Waste Else, the liquid moves in the another way.

## Return type

bool

## getWCValveStateNode()

This method returns the id of the valve state variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the ValveState variable in the OPC UA Address Space.

## Return type

opcua.Node

## setWCValveState(state)

It sets the state of the valve.

#### Parameters

**state** (*bool*) – True means fluid from Collect to Waste. Else, the fluid moves in the another way.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## startManualControl()

This function allows the user to reach the control of the machine from this Python Client. It starts the manual control when the machine gives back a confirmation.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## stopAll()

This function stops the manual control of the Vapourtec R-Series Flow Chemistry Machine.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

#### 2.1.2.1.1.9 mfc

## class MFCType(root: Node)

A MFCType object contains all of the methods and object to monitor and control a MFC Module in the R-Series Controller.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the R2 object id in the OPC UA Address Space.

## **Type**

opcua.Node

## erroState

It is the ErrorState variable id in the OPC UA Address Space.

## **Type**

opcua.Node

#### gasType

It is the GasType variable id in the OPC UA Address Space.

## Type

opcua.Node

#### flowRate

It is the FlowRate variable id in the OPC UA Address Space.

#### Type

opcua.Node

## getErrorState()

It reports if there was an error with the module.

#### **Returns**

It is True if there was an error. Otherwise, it si False.

#### **Return type**

bool

#### getErrorStateNode()

This method returns the id of the ErrorState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the ErrorState variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getFlowRate()

This function gives the current pump flow rate.

#### Returns

It is the pump flow rate in ul/min.

#### Return type

integer

## getFlowRateNode()

This method returns the id of the flow rate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

It is the id of the flowRate variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getGasType()

It gives the gas type MFC has set.

#### Returns

This is the code for the gas type.

#### Return type

integer

## getGasTypeNode()

This method returns the id of the ErrorState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the ErrorState variable in the OPC UA Address Space.

opcua.Node

## setFlowRate(flowRate)

It sets the flow rate to the pump. When flow rate is setted in 0, the pump is turned off. If the pump is setted in other value than 0, the pump is immediatly turned on. The flow rate must be set in a suitable value for the connected pump in MFC. In other case, the flow rate will not be setted in the machine.

#### **Parameters**

**flowRate** (*integer*) – It is the pump flow rate in ul/min.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

#### 2.1.2.1.1.10 module

## class ModuleType(root)

This is a base class for class of objects that delivers some type of reagent and/or solvent.

#### getPumps()

It gives the pumps dictionary.

#### Returns

It is the list of pumps.

## **Return type**

dict of *PumpType* 

## 2.1.2.1.1.11 pump

## class PumpType(root)

A PumpType object contains all of the methods and variables to control and monitor a pump inside an R2Module.

This object can set and get the values of their attributes members but cannot subscribe them. In order to subscribe, the variable id must to be getted from inside the member object.

In example, to subscribe the flow rate, the variable id has to be getted from:

PumpType.flowRate.getFlowRateNode().

## **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the id of the pump object in the OPC UA Address Space.

#### **Type**

opcua.Node

## flowRate

It is the id of flow rate variable in the OPC UA Address Space.

#### **Type**

opcua.Node

#### srValveState

It is the id of the state of the valve in the OPC UA Address Space.

## **Type**

opcua.Node

#### ilValveState

It is the id of the state of the valve in the OPC UA Address Space.

#### **Type**

opcua.Node

#### getFlowRate()

This function gives the current pump flow rate.

#### Returns

It is the pump flow rate in ul/min.

#### Return type

integer

## getFlowRateNode()

This method returns the id of the flow rate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the flowRate variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getILValveState()

This gives the state of the inject or load selection valve.

## Returns

Depends on which is teh valve, true means liquid from Inject to Load. Else, the liquid moves in the another way.

#### Return type

bool

## getILValveStateNode()

This method returns the id of the ILValveState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

It is the id of the ILValveState variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getSRValveState()

This gives the state of the Solvent or Reagent selection valve

#### Returns

True means liquid from Reagent Else, the liquid moves in the another way.

## Return type

bool

## getSRValveStateNode()

This method returns the id of the SRValveState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the SRValveState variable in the OPC UA Address Space.

#### Return type

opcua.Node

## setFlowRate(flowRate)

It sets the flow rate to the pump. When flow rate is setted in 0, the pump is turned off. If the pump is setted in other value than 0, the pump is immediatly turned on. The flow rate must be set in a suitable value for the connected pump in R2. In other case, the flow rate will not be setted in the machine.

#### **Parameters**

**flowRate** (*integer*) – It is the pump flow rate in ul/min.

## Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

#### setILValveState(state)

It sets the state of the inject or load selection valve.

#### **Parameters**

**state** (*bool*) – Depends on which is the valve, true means fluid from Inject to Load. Else, the fluid moves in the another way.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

## setSRValveState(state)

It sets the state of the Solvent or Reagent selection valve.

#### **Parameters**

**state** (*bool*) – Depends on which is the valve, true means fluid from Reagent to Solvent Else, the fluid moves in the another way.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

bool

#### 2.1.2.1.1.12 r2

#### class R2Type(root)

A R2Type object contains all of the methods and object to monitor and control a R2 Module in the R-Series Controller.

## **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the R2 object id in the OPC UA Address Space.

## **Type**

opcua.Node

## maxPressureLimit

It is the id of the maxPressureLimit variable in the OPC UA Address Space. This contains information about the maximum pressure limit allowed for the pump.

#### Type

opcua.Node

## pressureLimit

It is the pressureLimit variable id in the OPC UA Address Space. This is the current pressure limit set point in system.

## **Type**

opcua.Node

#### pumpsPressure

It is the measured pumpsPressure variable id in the OPC UA Address Space.

## Type

opcua.Node

## pumpsPerformance

It is the id of PumpsPerformance variable in the OPC UA Address Space.

#### **Type**

opcua.Node

## pump

It is a dictionary of pumps. Each dictionary entry contains the methods, object and variable necesary to control and monitor a pump. The pumps are alphabetic labeled, so the key of the dict are ['A', 'B'].

#### Type

dict of PumpType

## getMaxPressureLimit()

It gives the maximum pressure limit setted in the module.

#### Returns

It is the pressute limit in mbar.

## Return type

float

#### getMaxPressureLimitNode()

This method returns the id of the max pressure limit variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the maxPressureLimit variable in the OPC UA Address Space.

#### **Return type**

opcua.Node

## getPressureLimit()

It gives the pressure limit value.

#### Returns

It is the actual pressure limit.

#### Return type

float

## getPressureLimitNode()

This method returns the id of the pressure limit variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### **Returns**

It is the id of the pressureLimit variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getPumpA()

It returns the pump A object.

#### Returns

It is the the object that contains the methods, objects and variables to control and monitor the Pump A.

## Return type

PumpType

## getPumpB()

It gives the pump B object.

#### Returns

It is the object that contains the methods, objects and variables to control and monitor the Pump B.

## Return type

PumpType

#### getPumpsPerformance()

It returns the performance of all of the pumps.

#### **Returns**

It is a list with the performance of each pump. [0] is the pumpA performance [1] is the pumpB performance

## Return type

list

## getPumpsPerformanceNode()

This method returns the id of the pumps performance variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of PumpsPerformance variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getPumpsPressures()

It returns the pressures of the pumps and the R2 Module.

#### Returns

It is a list with the pressure each pump and the system. list()[0] is the module Pressure list()[1] is the pumpA pressure list()[2] is the pumpB pressure

## Return type

list

#### getPumpsPressuresNode()

This method returns the id of the pumps pressures variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the PumpsPressures variable in the OPC UA Address Space.

## Return type

opcua.Node

## setMaxPressureLimit(pressure)

Sets the maximum pressure admited in the pumps by the system

#### **Parameters**

**pressure** (*float*) – It is the new maximum pressure limit for all of the pumps.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

bool

## setPressureLimit(pressure)

It sets the pressure limit value.

#### **Parameters**

pressure (float) - It is the new pressure to set.

#### **Returns**

It is true if the value set was successfully aplied.

#### **Return type**

bool

## 2.1.2.1.1.13 r4

## class R4Type(root)

The R4Type is an object that contains all the methods, variables and objects that contains the information to control and monitor the R4Modules.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the id of the R4 object in the OPC UA Address Space.

## **Type**

opcua.Node

## reactor

It is a dictionary that contains all of the reactors. Each R4 has 4 reactors. The key of each reactor is a number from 1 to 4.

## **Type**

dict of ReactorType

## getReactors()

It return a dictionary with all the reactors in R4 module.

#### Returns

This has all the reactors.

## **Return type**

dict

#### 2.1.2.1.1.14 reactor

## class ReactorType(root)

This ReactorType class represent each reactor that has present in the R-Series Controller. This contains all the methods and variables with information about one reactor.

## **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

It is the id of the Reactor in the OPC UA Address Space.

## **Type**

opcua.Node

#### temperature

This is the id of the temperature measurement in the OPC UA Address Space.

#### **Type**

opcua.Node

#### power

This is the id of the power variable in the OPC UA Address Space.

#### **Type**

opcua.Node

#### reactorType

It is the id of the reactor type variable in the OPC UA Address Space.

#### **Type**

opcua.Node

#### class ReactorTypeEnum

The posible types of reactor founded HEATED\_NOR COOLED\_TUBE COOLED\_COLUMN MIDRANGE\_TUBE UV150 NIK\_BATCH CHIP\_REACTOR EC\_REACTOR

#### getPower()

It returns the power of the Reactor.

## Returns

The reactor power.

## Return type

float

## getPowerNode()

This method returns the id of the reactor's power. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

This is the id of the power variable in the OPC UA Address Space.

#### Return type

opcua.Node

## getReactorType()

It returns the Reactor Type in an integer. This is defined by self.ReactorTypeEnum

## Returns

It is the reactor type.

## Return type

ReactorTypeEnum

### getReactorTypeNode()

This method returns the id of the Reactor Type variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the reactor type variable in the OPC UA Address Space.

#### Return type

opcua.Node

### getTemperature()

It gets the last temperature measuremt of the reactor.

#### Returns

This is the measured temperature. A -1000 value means the reactor is turned off.

#### Return type

float

### getTemperatureLimits()

It returns the temperature limits for the set point.

#### Returns

It is a list that contains the low and high limits.

### Return type

list

### getTemperatureLimitsNode()

This method returns the id of the tempearture limits. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the Temperature Limits variable in the OPC UA Address Space.

### Return type

opcua.Node

## getTemperatureNode()

This method returns the id of the Temperature variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the Temperature variable in the OPC UA Address Space.

#### Return type

opcua.Node

### setTemperature(temp)

It sets the temperature set point. This temperature will be reached when the StartManualControl method is run.

The temperature will be set into the limits. If a lower or higher target temperature is send, the closer limit will be set

#### **Parameters**

**temp** (*integer*) – This is the new temperature set point.

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

### 2.1.2.1.1.15 sf10

### class SF10Type(root)

The SF10Type class implements the low level functions for controlling the components of the R-Series Machine.

This class allows the user to write and run scripts for custom automation routines.

#### **Parameters**

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

This is the parent node in the OPC UA Address Space.

## Type

opcua.Node

#### maxPressureLimit

This is the MaxPressureLimit node in the OPC UA Address Space.

### **Type**

opcua.common.node.Node

## pressureLimit

This is the PressureLimit node in the OPC UA Address Space.

#### **Type**

opcua.common.node.Node

## bprRegulationPressure

This is the BPRRegulationPression node in the OPC UA Address Space.

### pumpPressure

This is the PumpPressure node in the OPC UA Address Space.

#### **Type**

opcua.common.node.Node

## gasFlowRate

This is the GasFlowRate node in the OPC UA Address Space.

### **Type**

opcua.common.node.Node

#### flowrate

This is the FlowRate node in the OPC UA Address Space.

#### Type

opcua.common.node.Node

#### **srValveState**

This is the SRValveState node in the OPC UA Address Space.

#### Type

opcua.common.node.Node

### getBPRRegulationPressionNode()

This method returns the id of the SF10 BPRRegulationPression variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 BPRRegulationPression variable in the OPC UA Address Space.

### Return type

opcua.Node

### getBPRRegulationPressure()

It gives the current pressure set for Back Pressure Regulation mode.

#### Returns

It is the current pressure set.

#### Return type

float

### getFlowRate()

It gives the current flow rate for constant flow rate operation mode.

#### Returns

It is the current flow rate.

#### Return type

float

## getFlowRateNode()

This method returns the id of the FlowRate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 FlowRate variable in the OPC UA Address Space.

#### Return type

opcua.Node

### getGasFlowRate()

It gives the current flow rate for gas operation mode

#### Returns

It is the current flow rate.

#### Return type

float

## getGasFlowRateNode()

This method returns the id of the SF10 GasFlowRate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 GasFlowRate variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getMaxPressureLimit()

It gives the actual maximum pressure limit.

### Returns

It is the actual maximum pressure limit.

## Return type

float

#### getMaxPressureLimitNode()

This method returns the id of the SF10 MaxPressureLimit variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 MaxPressureLimit variable in the OPC UA Address Space.

## **Return type**

opcua.Node

### getOperationMode()

It gives the current operation mode set on SF10 pump.

### Returns

It is the current operation mode 0 means Constant flow rate mode 1 means Pressure Regulation mode 2 means volume dose regulation mode 3 means ramp flowrate mode 4 means gas mode 5 means Oscillation Mode

#### **Return type**

integer

### getOscillation()

It gives the parameters set for the SF10 Oscillation mode.

#### Returns

It returns a list with the speedul and displacement parameters.

### Return type

list

### getPressureLimit()

It gives the current pressure limit set on the device.

#### Returns

It is the current pressure limit

### Return type

float

### getPressureLimitNode()

This method returns the id of the SF10 PressureLimit variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

### Returns

This is the id of the SF10 PressureLimit variable in the OPC UA Address Space.

## Return type

opcua.Node

### getPumpPressure()

It gives the current pressure on the pump

## Returns

It is the pump pressure

## Return type

float

#### getPumpPressureNode()

This method returns the id of the SF10 Pumppressure variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

### Returns

This is the id of the SF10 PumpPressure variable in the OPC UA Address Space.

### Return type

opcua.Node

#### getSRValveState()

It gives the current state for the solvent-reagent selection valve.

#### Returns

This is the current valve state. True means the pump delivers reagent. False means the pump delivers solvent.

### **Return type**

bool

## getSRValveStateNode()

This method returns the id of the SRValveState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the SF10 SRValveState variable in the OPC UA Address Space.

### Return type

opcua.Node

### setBPRRegulationPressure(newPressure)

It sets the pressure for the Back Pressure Regulation mode.

#### **Parameters**

**newPressure** (*float*) – It is the new pressure for the regulator.

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### setFlowRate(newFlowRate)

It sets the flow rate for constant flow rate mode.

### **Parameters**

**newFlowRate** (*float*) – It is the new flow rate.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

## setGasFlowRate(newFlowRate)

It sets the flow rate for gas mode operation

#### **Parameters**

**newFlowRate** (*float*) – It is the new flow rate

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

#### setMaxPressureLimit(newMaxPressureLimit)

It sets a new maximum pressure limit.

#### **Parameters**

**newMaxPressureLimit** (*float*) – It is the new maximum pressure limit.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### **Return type**

bool

### setOperationMode(newMode)

It sets the Pump Mode to Pressure Regulation. You can switch between Flow, Pressure Regulation (REG), DOSE, RAMP, GAS.

#### **Parameters**

**newMode** (*integer*) – It is the new operation mode. 0 means Constant flow rate mode 1 means Pressure Regulation mode 2 means volume dose regulation mode 3 means ramp flowrate mode 4 means gas mode 5 means Oscillation Mode

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### setOscillation(speedul, displacement)

It configures the oscillation parameters for SF10 scillation mode operation.

#### **Parameters**

- **speedul** (*integer*) It is the speed at which the pump oscillates
- displacement (integer) It is the amount of volume that is oscillated back and forth

#### **Returns**

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

## setPressureLimit(newPressureLimit)

It sets the pressure limit for SF10 operation.

#### **Parameters**

**newPressureLimit** (*float*) – It is the new pressure limit

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## Return type

bool

#### setSRValveState(newState)

It sets the state of the solvent or reagent selection valve.

### **Parameters**

**newState** (bool) – This is the new valve state. True means the pump will deliver reagent. False means the pump will deliver solvent.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### **Return type**

bool

## 2.1.2.1.1.16 syringe

#### class SyringePumpType(root)

SyringePumpType class is an object to monitor and control external syringe pumps.

#### **Parameters**

**root** (*opcua.Node*) – It is the id of this node in the OPC UA Address Space.

#### root

It is the OPC UA Address Space information about this object.

#### Type

opcua.Node

### valveState

It is the id of the ValveState variable in the OPC UA Address Space.

#### Туре

opcua.Node

#### position

It is the id of the Position variable in the OPC UA Address Space.

#### Type

opcua.Node

### flowRate

It is the id of the FlowRate variable in the OPC UA Address Space.

## Type

opcua.Node

#### getErrorState()

It reports if there was an error with the module.

## Returns

It is True if there was an error. Otherwise, it is False.

## Return type

bool

#### getErrorStateNode()

This method returns the id of the ErrorState variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

It is the id of the ErrorState variable in the OPC UA Address Space.

## Return type

opcua.Node

#### getFlowRate()

This function gives the current pump flow rate.

#### Returns

It is the pump flow rate in ul/min.

### **Return type**

integer

#### getFlowRateNode()

This method returns the id of the flow rate variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the flowRate variable in the OPC UA Address Space.

#### Return type

opcua.Node

### getPosition()

It gives the syringe position

#### **Returns**

It is the syringe position

### Return type

integer

### getPositionNode()

This method returns the id of the Position variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the Position variable in the OPC UA Address Space.

### Return type

opcua.Node

## getValveState()

This gives the state of the valve

#### Returns

Depends on which is teh valve, true means liquid from Reagent to Solvent, Collect to Waste or Inject to Load. Else, the liquid moves in the another way.

### Return type

bool

#### getValveStateNode()

This method returns the id of the valve state variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

It is the id of the ValveState variable in the OPC UA Address Space.

#### **Return type**

opcua.Node

### setFlowRate(flowRate)

It sets the flow rate to the pump. When flow rate is setted in 0, the pump is turned off. If the pump is setted in other value than 0, the pump is immediatly turned on. The flow rate must be set in a suitable value for the connected pump in R2. In other case, the flow rate will not be setted in the machine.

#### **Parameters**

**flowRate** (*integer*) – It is the pump flow rate in ul/min.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### setPosition(newPosition)

It sets a new position for the syringe

### **Parameters**

**newPosition** (*integer*) – It is he new position to set.

#### setValveState(state)

It sets the state of the valve.

#### **Parameters**

**state** (*bool*) – Depends on which is the valve, true means fluid from Reagent to Solvent, Collect to Waste or Inject to Load. Else, the fluid moves in the another way.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

### 2.1.2.1.1.17 uv150

#### class UV150Type(root)

The UV150Type class instantiates UV150 objects. This objects allows the user to turn the UV lamp on and modify its power.

### Parameters

root (opcua. Node) – It is the parent node in the OPC UA Address Space.

#### root

This is the id of the UV150 in the OPC UA Address Space.

### **Type**

opcua.Node

### lampPower

This is the id of the lamp power in the OPC UA Address Space.

## Type

opcua.Node

### getLampPower()

It gets the UV150 lamp power

#### Returns

This returns an integer that is lamp power. The lamp power is a percentage number.

### **Return type**

integer

## getLampPowerNode()

This method returns the id of the Lamp Power. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

## Returns

This is the id of the Lamp Power in the OPC UA Address Space. This id allows to subscribe the variable to monitor it.

### Return type

opcua.Node

## setLampPower(power)

It sets the UV150 lamp power

### **Parameters**

**power** (*integer*.) – This is the new lamp power. The lamp power is a percentage number, but only can be between 50 and 100. To turn the lamp off, power needs to be 0.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### 2.1.2.1.1.18 vbfr

### class VBFRType(root)

A VBFRType Object contains all the variables and methods to know the measurements and information about the Variable Bed Flow Reactor

#### **Parameters**

root (opcua. Node) – It is the id of the VBFR object in the OPC UA Address Space.

#### root

It is the id of the VBFR object in the OPC UA Address Space.

### **Type**

opcua.Node

### position

It is the id of the VBFR Piston Position variable in the OPC UA Address Space.

#### **Type**

opcua.Node

#### dPress

It is the id of the VBFR Differential Pressure variable in the OPC UA Address Space.

#### **Type**

opcua.Node

#### volume

It is the id of the VBFR Volume variable in the OPC UA Address Space.

#### **Type**

opcua.Node

### getDifferentialPressure()

It gives the value of the VBFR differential pressure in mBar.

#### Returns

This is the differential pressure.

## Return type

integer

## getDifferentialPressureNode()

This method returns the id of the VBFR Differential Pressur variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the VBFR Differential Pressure variable in the OPC UA Address Space.

### Return type

opcua.Node

## getPosition()

It gives the Piston Position in mm.

#### Returns

It is the position of the piston.

### **Return type**

float

## getPositionNode()

This method returns the id of the VBFR Piston Position variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the VBFR Piston Position variable in the OPC UA Address Space.

### Return type

opcua.Node

### getVolume()

It gives the inyected or loaded volume from the VBFR in mm<sup>3</sup>

#### Returns

It is the VBFR volume.

#### Return type

float

### getVolumeNode()

This method returns the id of the VBFR Volume variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in reading Data Example.py

#### Returns

This is the id of the VBFR Volume variable in the OPC UA Address Space.

#### Return type

opcua.Node

### 2.1.2.1.2 experiment package

### 2.1.2.1.2.1 experiment

## class ExperimentType(root)

ExperimentType class is an object that contains the high level commands to work with the R-Series machine.

It has the experiment settings information and the reaction manager methods to command the list of reactions that are running in the system.

Also, it gives statistics about the experiments.

#### **Parameters**

root (opcua. Node) – It is the id of this node in the OPC UA Address Space.

#### root

It is the id of the experiment object in the OPC UA Address Space.

### **Type**

opcua.Node

### experimentSetup

It is the id of the experiment setup object in the OPC UA Address Space.

#### Type

*ExperimentSetupType* 

### reactionManager

It is the id of the reaction manager object in the OPC UA Address Space.

#### Type

ReactionManagerType

### statistics

It is the id of the experiment statistics in the OPC UA Address Space.

#### Type

ExperimentStatisticType

## getExperimentSetup()

It returns the experiment setup object.

#### **Returns**

It is the setup object.

## Return type

 ${\it Experiment Setup Type}$ 

## getReactionManager()

It gives the Reaction Manager object.

#### Returns

It is the Reaction Manager object.

## Return type

Reaction Manager Type

## getStatistics()

It gives the Experiment Statistics object

### Returns

It is the Statistics object.

## Return type

ExperimentStatisticsType

### 2.1.2.1.2.2 experimentSetup

## class ExperimentSetupType(root)

The ExperimentSetupType class contain the methods that allow the user to save and restore the System settings.

#### **Parameters**

root (opcua. Node) – It is the experiment setup object id in the OPC UA Address Space.

#### root

It is the id of the experimentSetup object in the OPC UA Address Space.

### **Type**

opcua.Node

## getExperiment(pathToSave=None)

It gives the experiment data. The data returned could be saved in a file and then it could be send by loadExperiment.

#### Returns

This is the experiment data.

### Return type

**JSON** 

### getSystemSettings()

It returns the settings of all the components presents in the loaded experiment on the R-Series Controller.

#### Returns

These are the system settings.

### **Return type**

**JSON** 

### loadExperiment(experiment)

It loads an experiment into the system. The experiment data to send is getted by getExperiment method.

#### **Parameters**

**experiment** (*JSON*) – It is the experiment to load.

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

## **Return type**

bool

## loadSystemSettings(settings)

It loads all the system settings in the current experiment on the R-Series Controller.

#### **Parameters**

**settings** (*JSON*) – It is the system settings.

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### 2.1.2.1.2.3 experimentStatistics

### class ExperimentStatisticType(root)

An ExperimentStatisticsType object contains the methods to obtain the statistics information about the loaded experiment.

#### **Parameters**

root (opcua.Node) – It is the id of the ExperimentStatistics in the OPC UA Address Space.

#### root

It is the id of the ExperimentStatistics in the OPC UA Address Space.

### **Type**

opcua.Node

### getExperimentGraphs(filename=None)

It gives the graphs from the R-Series Controller. This graph could be stored in a file or could be returned as a list.

#### **Parameters**

**filename** (*string*, *optional*) – It is the path where the file will be stored. The default is None.

#### **Returns**

**rspList** – It is the list with all graphs.

If a filename is given, the response of this method is None and the graphs will be stored in a file.

### Return type

list of list of string or None.

### getReactionStatistics(reaction)

It gives the temporal statistic of a given reaction.

#### **Parameters**

reaction (integer) – It is the index of the reaction in ReactionList

### Returns

These are the statistics of the selected reaction. If the reaction is not in the RactionList, it returns None.

### Return type

JSON or None.

### 2.1.2.1.2.4 reactionManager

### class ReactionManagerType(root)

The Reaction Manager class allows the user manage the Reaction List.

Reactions can be added, deleted or desabled from the Reaction List. Then, this list could be started or stoped with the methods provides from an object of this class.

### **Parameters**

**root** (opcua. Node) – It is the Reaction Manager's id in the OPC UA Address Space.

#### root

It is the Reaction Manager's id in the OPC UA Address Space.

## Type

opcua.Node

## reactionStatus

It is the ReactionStatus variable id in the OPC UA Address Space.

#### **Type**

opcua.Node

### addReactionToList(reaction)

This method add one reaction to the current Reaction List.

#### **Parameters**

**reaction** (*String*) – This is the reaction to add. The reaction must be the followings field, in a CSV format.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### **Return type**

bool

#### deleteReaction(reaction)

It deletes a given reaction.

#### **Parameters**

**reaction** (integer) – This is the index of the reaction to be removed of the Reaction List.

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

#### Return type

bool

### disableReaction(reaction)

This method disables a given reaction. When a reaction is disabled, it will not be runned when startReaction is called.

#### **Parameters**

**reaction** (integer) – This is the index of the reaction to be disabled in the Reaction List.

### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

## ${\tt enable Reaction}(\textit{reaction})$

This method enables a given reaction. When a reaction is enabled, it will be runned when startReaction is called.

### **Parameters**

**reaction** (integer) – This is the index of the reaction to be enabled in the Reaction List.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### getCurrentReactionIndex()

It returns the position of the current reaction in the reaction list.

### Returns

It is the index of the reaction.

#### **Return type**

integer

### getCurrentReactionStatus()

It gives the status of the current reaction. It could be one of the following states:

IDLE = 0

NOT\_RUNNING = 1

 $WAIT_RACK_CHANGE = 2$ 

 $SET_PREHEATING = 3$ 

HEATING = 4

 $WAIT4PRE_WASH = 5$ 

 $PUMP_DEAD_VOLUME = 6$ 

WAIT4DEAD\_VOLUME = 7

 $WAIT4SAMPLE\_LOOP\_LOADS = 8$ 

 $START_REACTION = 9$ 

 $RUNNING_REACTION = 10$ 

WAIT4COLLECTION = 11

POST\_WASHING, = 12

 $WAIT4POST_WASH$ , = 13

FINAL\_COLLECTION, = 14

WAIT4FINAL\_COLLECTION, = 15

CHECK4REACTIONS, = 16

CLEANING, = 17

WAIT4CLEANING, = 18

COOLING, = 19

WAIT4COOLING, = 20

 $STOP_REACTION = 21$ 

#### Returns

It is the current reaction status.

#### **Return type**

integer

## getEnableReactionStatus(index)

It reports if a given reaction is enabled to be automatically excecuted or not.

### **Parameters**

**index** (*integer*) – It is the 0-based position of the desired reaction in the reaction list

#### Returns

It is True when the reaction is enabled.

## Return type

bool

## getReactionList()

It gives the reaction list of the actual experiment file loaded. It needs to be an experiment set and loaded a reaction list or created before to be meaningful. The reaction list is in CSV Format. It is received the string separator and the decimal point.

#### Returns

It is the reaction list in CSV format. The first line is the table's header. The following lines are the fields. The lines are delimited by 'n'.

### **Return type**

String

### getReactionParameters(position=0)

It returns all the parameteres of a given reaction in the Reaction List.

#### **Parameters**

**position** (integer) – It is the index of the reaction in the reaction list

#### Returns

These are the reaction list parameters, including CSV string separators, CSV deciman point, the information about all of the pumps and reactors of each reaction in the Reaction List and the amount of reactions. It could be direct loaded to a JSON object.

### Return type

String

## getReactionStatusNode()

This method returns the id of the reaction status variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in readingDataExample.py

#### Returns

This is the id of the reaction status variable in the OPC UA Address Space.

### **Return type**

opcua.Node

### loadReactionList(reactionList)

loadReactionList replaces the actual reaction list with a new one. This method needs an experiment set before its execution.

#### **Parameters**

**reactionList** (*String*) – It is a CSV format string with the reaction list to load. It has to be a valid Vapourtec RSeries reaction table.

### Return type

None.

### resetReactionList()

This method deletes all of the reactions in the current Reaction List.

#### Return type

None.

#### setReactionPosition(curPos, newPos)

setReactionPosition changes the position of a Reaction in the Reaction List.

#### **Parameters**

- **curPos** (*integer*) This is the current position of the reaction.
- **newPos** (*integer*) This is the position where the reaction will be placed.

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### startReaction()

startReaction sends a command to run all of the enabled reactions in the current Reaction List

#### Returns

It is true when the method was applied successfully to the system. Otherwise, it is false.

### Return type

bool

### stopReaction()

stopReaction sends a command to stop the current reaction execution.

### Return type

None.

### 2.1.2.1.3 identification package

### 2.1.2.1.3.1 identification

#### class IdentificationType(root)

IdentificationType is an object that gives the machine information. It has a opc Node that contains its opc address. Also has two components that gives the software and hardware information. The data could be accessed by itself, by calling this class methods or by calling each component methods.

### **Parameters**

root (opcua.Node) – It is the id of this node in the OPC UA Address Space.

## root

This has the OPC-UA information.

### **Type**

opcua.Node

### equipmentIdentification

It contains the hardware information.

#### **Type**

EquipmentIdentificationType

### softwareIdentification

It has the software data.

### **Type**

SoftwareIdentificationType

## getEquipmentIdentifier()

This function gives the name of the machine.

#### Returns

This machine is an RSeries Flow Chemistry.

### Return type

String

### getEquipmentManufacturer()

This function returns Vapourtec. This is the builder company of the machine.

#### Returns

Vapourtec design and made the RSeries machine.

## Return type

String

### getHardwareRevision()

This funtion gives the hardware version.

#### Returns

This is the hardware version. The format of the harware revision is vXX.YY.ZZ, where:

- v is an arbitrary character
- XX is a major change
- YY is a minor change
- ZZ is a patch

## Return type

String

## getSoftwareDeveloper()

This function gives the name of the software developer company.

#### Returns

Developer name.

## Return type

String

## getSoftwareIdentifier()

This function gives the human readable name of the software at the controller.

#### **Returns**

This is the software name identifier.

### Return type

String

## getSoftwareVersion()

This function gives the version of the software at the controller.

#### Returns

This is the software version. The format of the software version is vXX.YY.ZZ, where:

- v is an arbitrary identifier
- XX is a major change.
- YY is a minor change.
- ZZ is a patch.

## Return type

String

## 2.1.2.1.3.2 equipmentIdentification

## class EquipmentIdentificationType(root)

An EquipmentIdentificationType object has the hardware identification information.

### **Parameters**

root (opcua. Node) – It is the id of this node in the OPC UA Address Space.

#### root

It is the OPC-UA Address Space of this object

### **Type**

opcua.Node

#### harwareRevision

It is a OPC-UA variable. It has the hardware revision data in a String value.

#### Type

opcua.Node

### equipmentIdentifier

It is a OPC-UA variable. It has the equipment identifier data in a String value.

#### Type

opcua.Node

### equipmentManufacturer

It is a OPC-UA variable. It contains the equpiment manufacturer data in a String value.

#### Type

opcua.Node

## equipmentIdentifier()

This gives the name of the machine.

#### Returns

This machine is an RSeries Flow Chemistry.

### Return type

String

### equipmentManufacturer()

This function returns Vapourtec. This is the buider company of the machine.

#### Returns

Vapourtec design and made the RSeries machine.

## Return type

String

#### hardwareRevision()

This gives the hardware version.

## Returns

This is the hardware version. The format of the harware revision is vXX.YY.ZZ, where :

- v is an arbitrary character
- XX is a major change
- YY is a minor change
- ZZ is a patch

## Return type

String

### 2.1.2.1.3.3 softwareIdentification

## class SoftwareIdentificationType(root)

The SoftwareIdentificationType class contains information about the software development.

### **Parameters**

root (opcua. Node) – It is the id of this node in the OPC UA Address Space.

#### root

It is the OPC-UA Address Space information

## **Type**

opcua.Node

### softwareRevision

It is a OPC-UA variable. This contains the software revision version in a String value.

## Type

opcua.Node

#### softwareIdentifier

It is a OPC-UA variable. This variable contains the software identifier in a String value.

### **Type**

opcua.Node

## softwareDeveloper

It is a OPC-UA variable. It contains the software developer in a String value.

#### Type

opcua.Node

### softwareDeveloper()

This gives the name of the software developer company.

### Returns

Developer name

### Return type

String

### softwareIdentifier()

This gives a human readable name of the software at the controller.

#### Returns

This is the software name identifier.

#### **Return type**

String

### softwareVersion()

This gives the version of the software at the controller.

#### Returns

This is the software version. The format of the software version is vXX.YY.ZZ, where:

- v is an arbitrary identifier
- XX is a major change.
- YY is a minor change.
- ZZ is a patch.

## Return type

String

### 2.1.2.1.4 monitoring package

## 2.1.2.1.4.1 monitoring

## class MonitoringType(root)

MonitoringType class is an object to get information about the system status.

## **Parameters**

root (opcua. Node) – It is the id of this node in the OPC UA Address Space.

### root

It is the OPC-UA Address Space information about this object.

#### **Type**

opcua.Node

## rSeriesStaus

It is an object to get the status of the R-Series machine.

#### **Type**

StatusType

## getRSeriesStatus()

It gives the RSeries status object in this machine.

### Returns

It is the status node. This node will give access to call methods in StatusType.

#### Return type

StatusType

## 2.1.2.1.5 notification package

### 2.1.2.1.5.1 notification

### class NotificationType(root)

A NotificationType Object contains variables and methods related to equipment alerts.

#### **Parameters**

**root** (*opcua.Node*) – It is the NotificationType's id in the OPC UA Address Space.

#### root

It is the id of this object in the OPC UA Address Space.

### **Type**

opcua.Node

### getEquipmentAlerts()

It gives the equipment status.

The status could be one of the followings:

SYS OK = 0

 $SYS_OVERPRESSURE_TRIP = 1$ 

PUMPA\_OVERPRESSURE\_TRIP = 2

PUMPB\_OVERPRESSURE\_TRIP = 3

UNDERPRESSURE\_TRIP = 4

PUMPA\_UNDERPRESSURE\_TRIP = 5

PUMPB\_UNDERPRESSURE\_TRIP = 6

PUMPA LOGIC ERROR = 7

PUMPB\_LOGIC\_ERROR = 8

PUMPC\_OVERPRESSURE\_TRIP = 9

PUMPD\_OVERPRESSURE\_TRIP = 10

PUMPC UNDERPRESSURE TRIP = 11

PUMPD\_UNDERPRESSURE\_TRIP = 12

 $PUMPC\_LOGIC\_ERROR = 13$ 

 $PUMPD_LOGIC_ERROR = 14$ 

#### Returns

It is the sttatus of the equipment

## Return type

integer

## getEquipmentAlertsNode()

This method returns the id of the equipment alerts variable. This id allows the user to subscribe the variable to a handler object.

To subscribe a variable, the handler class has to be defined and implement a datachange\_notification(self, node, value, data) function. This give the user the capability of monitoring the variable and makes a custom control of it.

Example of this will be found in reading Data Example.py

#### Returns

It is the id of the flowRate variable in the OPC UA Address Space.

### Return type

opcua.Node

### 2.1.2.1.6 base

#### class BaseNode

The BaseNode class has the common methods with the tasks that every device has to perform in order to get the right nodes on the server side.

### class NodeInfo(variableName, TagName, prototypeClass)

This is an auxiliar class that allows groups variables on a standard python container.

#### 2.1.2.1.7 factories

### class NodeFactory

The NodeFactory is a class of object that is provides the client the right type of member of a device class

```
getNode(node: Node, prototype: object = None) \rightarrow object | Node
```

The getNode method gives an object of proto instantiated with the node argument.

If proto is None, the node is returned.

### **Parameters**

- **node** (*Node*) This is the parameter of the prototype given.
- **prototype** (*object*, *optional*) This is the prototype to instantiate a new object, by default None

### Returns

This is the instantiated object of prototype. If prototype is None, the node will be returned.

## **Return type**

object | Node

### 2.1.2.1.8 reaction

### class Reaction

This class represents a Reaction in a Reaction List. Allows the user to load a Reaction and make copies with different configurations.

### class Collection

A collection is the way how reaction stores the final produt. It has an state, a diverterTime, a collectionTime, a number of vials, a total volume and a volume for each vial. A Reaction has information of one Collection.

## class Pump

A Pump has Flowrate, Volume Ratio, Concentration Ratio, Quantity, Advance Retard and Regent Concentration. It depends of the kind of Pump the experiment has, the valid changes of Volume or Conentration Ratio. The time which the pump is working is a function of flowrate and quantity selected. A Reactor has 8 Pumps, stored in a list numbered from 1 to 8.

#### class Reactor

A reactor has a tempreature and a residence time. Each Reaction has 8 reactors, numbered from 1 to 8.

### class UVReactor

An UVReactor is a kind of Reactor which has an extra UV power.

### class Wash

A wash is a time where a Pump is cleaning its system. A wash has a flowrate and a volume, which defines the wash time. A Reaction has two different times for wash before and after the Reaction is run.

## 2.1.3 utils

## get\_interest\_child(node, target)

Return the node's child that matchs with target string

#### **Parameters**

- **node** (opcua.Node()) Node that will be filtered
- target (string) string to match filtering the node

#### Returns

Node which browse name match with the string

### Return type

opcua.Node()

### infoAllChildren(node)

It prints all the browse name of the children of a given node

### **Parameters**

**node** (*opcua.Node*) – It is the parent node whose children will be printed.

### Return type

None.

### toLetter(num)

Convert an integer to equivalent capitalized alphabet letter.

### **Parameters**

**num** (*integer*) – It is the number to convert.

#### Returns

It is the capitalized alphabet letter

## **Return type**

char

## **PYTHON MODULE INDEX**

```
r
rseriesopc.client, 3
rseriesopc.model.base, 58
rseriesopc.model.devices.autosampler, 4
rseriesopc.model.devices.azura, 8
rseriesopc.model.devices.chiller, 10
rseriesopc.model.devices.detector, 14
rseriesopc.model.devices.factories, 16
rseriesopc.model.devices.flowChemistry, 16
rseriesopc.model.devices.ion, 18
rseriesopc.model.devices.manualControl, 20
rseriesopc.model.devices.mfc, 23
rseriesopc.model.devices.module, 25
rseriesopc.model.devices.pump, 25
rseriesopc.model.devices.r2, 28
rseriesopc.model.devices.r4,31
rseriesopc.model.devices.reactor, 31
rseriesopc.model.devices.sf10,34
rseriesopc.model.devices.syringe, 40
rseriesopc.model.devices.uv150,42
rseriesopc.model.devices.vbfr, 44
rseriesopc.model.experiment.experiment, 45
rseriesopc.model.experiment.experimentSetup,
rseriesopc.model.experiment.experimentStatistics,
rseriesopc.model.experiment.reactionManager,
rseriesopc.model.factories, 58
rseriesopc.model.identification.equipmentIdentification,
rseriesopc.model.identification.identification,
rseriesopc.model.identification.softwareIdentification,
rseriesopc.model.monitoring.monitoring, 56
rseriesopc.model.notification.notification,
rseriesopc.model.reaction, 58
rseriesopc.utils, 59
```

62 Python Module Index

# **INDEX**

| A  | equipmentIdentifier (EquipmentIdentificationType  |
|--|---|
| <pre>addReactionToList() (ReactionManagerType<br/>method), 49</pre>  | attribute), 54 equipmentIdentifier() (EquipmentIdentification-  |
| autoSampler (ManualControlType attribute), 20  | Type method), 54  |
| AutoSamplerType (class in rseriesopc.model.devices.autosampler), 4   | equipmentManufacturer (EquipmentIdentification-<br>Type attribute), 54  |
| AzuraPumpType (class in rseriesopc.model.devices.azura), 8   | equipmentManufacturer() (EquipmentIdentification-<br>Type method), 54<br>errorState (ChillerType attribute), 10   |
| В  | <pre>erroState (AzuraPumpType attribute), 8</pre>   |
| BaseNode (class in rseriesopc.model.base), 58<br>bprRegulationPressure (SF10Type attribute), 34  | erroState ( <i>MFCType attribute</i> ), 23<br>experiment ( <i>FlowChemistryType attribute</i> ), 17<br>experimentSetup ( <i>ExperimentType attribute</i> ), 46                      |
| C  | ExperimentSetupType (class in   |
| chiller (ManualControlType attribute), 20  | rseriesopc.model.experiment.experimentSetup),<br>47   |
| chillerType (ChillerType attribute), 10 ChillerType (class in rseriesopc.model.devices.chiller), 10  | ExperimentStatisticType (class in rseriesopc.model.experiment.experimentStatistics)   |
| <pre>cleanLoop() (AutoSamplerType method), 5 cleanNeedle() (AutoSamplerType method), 5 cleansSampleLoops() (AutoSamplerType method), 5 collectionValveState (AutoSamplerType attribute), 4 connect() (RSeriesClient method), 3</pre>   | ExperimentType (class in rseriesopc.model.experiment.experiment), 45  |
| current (IONType attribute), 18  | FlowChemistryType (class in   |
| D  | rseriesopc.model.devices.flowChemistry),  |
| deleteReaction() (ReactionManagerType method), 49 DetectorType (class in rseriesopc.model.devices.detector), 14 disableReaction() (ReactionManagerType method), 49   | flowRate (AzuraPumpType attribute), 8 flowRate (MFCType attribute), 23 flowRate (PumpType attribute), 25 flowrate (SF10Type attribute), 35 flowRate (SyringePumpType attribute), 40 |
| disconnect() (RSeriesClient method), 3, 4<br>dPress (VBFRType attribute), 44   | G   |
| E  | gasFlowRate (SF10Type attribute), 34 gasType (MFCType attribute), 23  |
| <pre>enableReaction() (ReactionManagerType method), 49 equipmentIdentification (IdentificationType at-</pre>   | <pre>get_interest_child() (in module rseriesopc.utils), 59 getAutoSampler() (ManualControlType method), 20 getBPRRegulationPressionNode() (SF10Type</pre>                           |
| EquipmentIdentificationType (class in rseriesopc.model.identification.equipmentIdentification.equipmen | method), 35 icgtivBRRegulationPressure() (SF10Type method), 35  |

| <pre>getChiller1() (ManualControlType method), 21 getChiller2() (ManualControlType method), 21 getChillers() (ManualControlType method), 21 getChillers() (ManualControlType method), 21</pre> getGasTypeNode() (MFCType method), 24 getHardwareRevision() (IdentificationType method), 21 53 | thod), |
|---|--------|
| getChillerType() (ChillerType method), 11 getIdentification() (FlowChemistryType method)  | thod)  |
| getChillerTypeNode() (ChillerType method), 11 17  | ,,     |
| getCollectionValveState() (AutoSamplerType getILValveState() (PumpType method), 26  |        |
| method), 5  getILValveStateNode() (PumpType method), 2  | 5      |
| getCollectionValveStateNode() (AutoSamplerType getLampPower() (UV150Type method), 43  |        |
| method), 5  getLampPowerNode() (UV150Type method), 43   |        |
| getCurrent() (IONType method), 18 getLampState() (DetectorType method), 14  |        |
| getCurrentNode() (IONType method), 18 getLampStateNode() (DetectorType method), 15  |        |
| getCurrentReactionIndex() (ReactionManagerType getLeaveRunning() (ChillerType method), 11   |        |
| method), 49 getLeaveRunningNode() (ChillerType method),   | 12     |
| <pre>getCurrentReactionStatus() (ReactionManager- getManualControl() (FlowChemistryType methor</pre>  |        |
| Type method), 50 getMaxPressureLimit() (AzuraPumpType meth  |        |
| getDetector() (ManualControlType method), 21 getMaxPressureLimit() (R2Type method), 28  | //, -  |
| getDifferentialPressure() (VBFRType method), 44 getMaxPressureLimit() (SF10Type method), 36   | )      |
| getDifferentialPressureNode() (VBFRType getMaxPressureLimitNode() (AzuraPun   |        |
| method), 44 method), 9  |        |
| <pre>getEnableReactionStatus() (ReactionManagerType getMaxPressureLimitNode() (R2Type method)</pre>   | 28     |
| method), 50 getMaxPressureLimitNode() (SF10Type metho   | 1), 36 |
| <pre>getEquipmentAlerts() (NotificationType method), 57  getMonitoring() (FlowChemistryType method),</pre>  | 17     |
| <pre>getEquipmentAlertsNode() (NotificationType getNode() (NodeFactory method), 58</pre>  |        |
| method), 57 getNotification() (FlowChemistryType method   | l), 17 |
| <pre>getEquipmentIdentifier() (IdentificationType getOperationMode() (SF10Type method), 36</pre>  |        |
| method), 53 getOscillation() (SF10Type method), 37  |        |
| <pre>getEquipmentManufacturer() (IdentificationType getPosition() (SyringePumpType method), 41</pre>  |        |
| method), 53 getPosition() (VBFRType method), 44   |        |
| <pre>getErrorState() (AzuraPumpType method), 8</pre> <pre>getPositionNode() (SyringePumpType method)</pre>  | 41     |
| <pre>getErrorState() (ChillerType method), 11</pre> <pre>getPositionNode() (VBFRType method), 45</pre>  |        |
| <pre>getErrorState() (MFCType method), 23</pre> <pre>getPower() (ReactorType method), 32</pre>  |        |
| <pre>getErrorState() (SyringePumpType method), 40</pre>   |        |
| <pre>getErrorStateNode() (AzuraPumpType method), 8</pre>  |        |
| <pre>getErrorStateNode() (ChillerType method), 11</pre> <pre>getPressureLimit() (SF10Type method), 37</pre>   |        |
| <pre>getErrorStateNode() (MFCType method), 23</pre> getPressureLimitNode() (R2Type method), 29  |        |
| <pre>getErrorStateNode() (SyringePumpType method), 40 getPressureLimitNode() (SF10Type method), 3</pre>   | 7      |
| <pre>getExperiment() (ExperimentSetupType method), 47  getPumpA() (R2Type method), 29</pre>   |        |
| <pre>getExperiment() (FlowChemistryType method), 17</pre> getPumpB() (R2Type method), 29  |        |
| <pre>getExperimentGraphs() (ExperimentStatisticType getPumpPressure() (SF10Type method), 37</pre>   |        |
| method), 48 getPumpPressureNode() (SF10Type method), 37   |        |
| <pre>getExperimentSetup() (ExperimentType method), 46  getPumps() (ModuleType method), 25</pre>   |        |
| <pre>getFlowRate() (AzuraPumpType method), 9</pre> <pre>getPumpsPerformance() (R2Type method), 29</pre>   |        |
| getFlowRate() (MFCType method), 24 getPumpsPerformanceNode() (R2Type method).   | 30     |
| getFlowRate() (PumpType method), 26 getPumpsPressures() (R2Type method), 30   |        |
| <pre>getFlowRate() (SF10Type method), 35</pre> getPumpsPressuresNode() (R2Type method), 36  | )      |
| getFlowRate() (SyringePumpType method), 40 getR4() (ManualControlType method), 21   |        |
| getFlowRateNode() (AzuraPumpType method), 9 getR4I() (ManualControlType method), 21   |        |
| getFlowRateNode() (MFCType method), 24 getR4II() (ManualControlType method), 21   |        |
|   | .1 1   |
| getFlowRateNode() (PumpType method), 26 getReactionList() (ReactionManagerType me   | thod), |
| <pre>getFlowRateNode() (SF10Type method), 35</pre> 51   |        |
| <pre>getFlowRateNode() (SF10Type method), 35 getFlowRateNode() (SyringePumpType method), 41 getReactionManager() (ExperimentType method)</pre>  | d), 46 |
| <pre>getFlowRateNode() (SF10Type method), 35</pre> 51   | d), 46 |

| <pre>getReactionStatistics() (ExperimentStatisticType</pre>  | method), 55  |
|--|--|
| method), 48  | harwareRevision (EquipmentIdentificationType at-   |
| <pre>getReactionStatusNode() (ReactionManagerType</pre>  | tribute), 54   |
| method), 51  | I  |
| getReactors() (R4Type method), 31  | 1  |
| getReactorType() (ReactorType method), 32  | identification (FlowChemistryType attribute), 16   |
| getReactorTypeNode() (ReactorType method), 32  | IdentificationType (class in   |
| getRSeries() (RSeriesClient method), 3, 4  | rseriesopc.model.identification.identification),   |
| getRSeriesStatus() (MonitoringType method), 56   | 52   |
| getSensorType() (ChillerType method), 12   | ilValveState ( <i>PumpType attribute</i> ), 26   |
| <pre>getSensorTypeNode() (ChillerType method), 12 getSetTemperature() (ChillerType method), 12</pre> | infoAllChildren() (in module rseriesopc.utils), 59   |
| getSetTemperature() (ChillerType method), 12 getSetTemperatureNode() (ChillerType method), 12        | IONType (class in rseriesopc.model.devices.ion), 18  |
| getSoftwareDeveloper() (IdentificationType method),  | isAReagentSourceModule() (ModuleTypeFactory  |
| 53   | method), 16  |
| <pre>getSoftwareIdentifier() (IdentificationType</pre>   | isConnected (RSeriesClient attribute), 3   |
| method), 53  | 1  |
| <pre>getSoftwareVersion() (IdentificationType method),</pre>   |  |
| 53   | lampPower (UV150Type attribute), 43  |
| <pre>getSRValveState() (PumpType method), 26</pre>   | lampState (DetectorType attribute), 14   |
| getSRValveState() (SF10Type method), 37  | leaveRunning (ChillerType attribute), 10   |
| <pre>getSRValveStateNode() (PumpType method), 27</pre>   | loadExperiment() (ExperimentSetupType method), 47  |
| getSRValveStateNode() (SF10Type method), 38  | loadLoopVolumeFromVial() (AutoSamplerType  |
| <pre>getStatistics() (ExperimentType method), 46</pre>   | method), 6   |
| <pre>getSystemSettings() (ExperimentSetupType method),</pre>   | <pre>loadReactionList() (ReactionManagerType method), 51</pre>                               |
| 47   | <u> </u>   |
| <pre>getTemperature() (ChillerType method), 13</pre>   | <pre>loadRSeries() (RSeriesClient method), 4 loadSystemSettings() (ExperimentSetupType</pre> |
| <pre>getTemperature() (ReactorType method), 33</pre>   | <pre>loadSystemSettings() (ExperimentSetupType</pre>   |
| <pre>getTemperatureLimits() (ReactorType method), 33</pre>   | lowerNeedleIntoPortA() (AutoSamplerType method),   |
| <pre>getTemperatureLimitsNode() (ReactorType method),</pre>  | 6  |
| 33   | <pre>lowerNeedleOverPortA() (AutoSamplerType method),</pre>                                  |
| <pre>getTemperatureNode() (ChillerType method), 13</pre>   | 6  |
| <pre>getTemperatureNode() (ReactorType method), 33</pre>   | <pre>lowerSyringePump() (AutoSamplerType method), 7</pre>                                    |
| getUV150() (ManualControlType method), 22  | 10 (10 of the table) (1 through the Type memory),  |
| <pre>getValveState() (SyringePumpType method), 41</pre>  | M  |
| <pre>getValveStateNode() (SyringePumpType method), 41</pre>  | makeReagentSourceType() (ModuleTypeFactory   |
| getVBFR() (ManualControlType method), 22   | method), 16  |
| getVoltage() (IONType method), 18  | manualControl (FlowChemistryType attribute), 17  |
| getVoltageNode() (IONType method), 19  | ManualControlType (class in  |
| getVolume() (VBFRType method), 45  | rseriesopc.model.devices.manualControl),   |
| getVolumeNode() (VBFRType method), 45  | 20   |
| getWavelengthAt() (DetectorType method), 15  | maxPressureLimit (AzuraPumpType attribute), 8  |
| getWavelengthNode() (DetectorType method), 15  | maxPressureLimit (R2Type attribute), 28  |
| getWavelengths() (DetectorType method), 15   | maxPressureLimit (SF10Type attribute), 34  |
| getWCValveState() (ManualControlType method), 22   | MFCType (class in rseriesopc.model.devices.mfc), 23  |
| <pre>getWCValveStateNode() (ManualControlType</pre>  | module   |
| goHome() (AutoSamplerType method), 5   | rseriesopc.client, 3   |
| <pre>goToPortA() (AutoSamplerType method), 6</pre>   | rseriesopc.model.base,58   |
| <pre>goToVialForCollection() (AutoSamplerType</pre>  | rseriesopc.model.devices.autosampler, 4  |
| method), 6   | <pre>rseriesopc.model.devices.azura, 8 rseriesopc.model.devices.chiller, 10</pre>            |
| 11   | rseriesopc.model.devices.detector, 14  |
| Н  | rseriesopc.model.devices.factories, 16   |
| hardwareRevision() (FauinmentIdentificationTyne  | 13c11c3opc.model.devices.lactories, 10   |

| rseriesopc.model.devices.flowChemistry,                           | P   |
|---|---|
| 16  | position (SyringePumpType attribute), 40  |
|   | position (VBFRType attribute), 44   |
|   | power (ReactorType attribute), 32   |
|   | pressureLimit (AzuraPumpType attribute), 8  |
|   | pressureLimit (R2Type attribute), 28  |
|   | pressureLimit (SF10Type attribute), 34  |
|   | primeSampleLoop() (AutoSamplerType method), 7   |
|   | pump (R2Type attribute), 28   |
|   | pumpPressure (SF10Type attribute), 34   |
|   | pumpsPerformance (R2Type attribute), 28   |
|   | pumpsPressure (R2Type attribute), 28  |
|   | PumpType (class in rseriesopc.model.devices.pump), 25                                 |
| rseriesopc.model.devices.uv150,42                                 | Tamp1)pe (ettiss in isertesope.model.devices.pump), 25                                |
| rseriesopc.model.devices.vbfr,44                                  | R   |
| recriseone model experiment experiment                            | R2Type (class in rseriesopc.model.devices.r2), 28                                     |
| 45  | na (Manual Control Type attribute) 20   |
| rseriesopc.model.experiment.experimentSetu                        | 14 (Manual Control Type autibute), 20   |
| 4/  | Donation (alagain namingon a model negation) 50                                       |
| rseriesopc.model.experiment.experimentStat                        | reaction (class in remesope. model. reaction), 36                                     |
| 48  |   |
| rseriesopc.model.experiment.reactionManage                        | rseriesopc.model.reaction), 58 Reaction.Pump (class in rseriesopc.model.reaction), 58 |
| 48  | Reaction. Reactor (class in recresope. model. reaction), 38                           |
| rseriesopc.model.factories,58                                     | ,   |
| rseriesopc.model.identification.equipmentI                        | rseriesopc.model.reaction), 58  |
| 54  |   |
| rseriesopc.model.identification.identifica                        | rseriesopc.model.reaction), 59 Reaction.Wash (class in rseriesopc.model.reaction), 59 |
| $J\omega$   |   |
| rseriesopc.model.identification.softwareId                        | leatification<br>ReactionManagerType (class in  |
| 55  | rseriesopc.model.experiment.reactionManager),   |
| <pre>rseriesopc.model.monitoring.monitoring,</pre>                | 48  |
| 56  |   |
| rseriesopc.model.notification.notification                        | leactionStatus (ReuchonmunagerType anriome), 49                                       |
| 57  | ReactorType (class in   |
| rseriesopc.model.reaction, 58                                     | rseriesopc.model.devices.reactor), 31   |
| rseriesopc.utils,59   | reactorType (ReactorType attribute), 32   |
|   | ReactorType (ReactorType Enum (class in   |
| <pre>ModuleType (class in rseriesopc.model.devices.module),</pre> | rseriesopc.model.devices.reactor), 32   |
| 25  | resetReactionList() (ReactionManagerType  |
| ModuleTypeFactory (class in                                       | method), 51   |
| rseriesopc.model.devices.factories), 16                           | riseSyringePump() (AutoSamplerType method), 7   |
| mone + orange (Lillour Classicatur Live a setterilar to) 16       | root (AutoSamplerType attribute), 4   |
|   | root (AzuraPumpType attribute), 8   |
|   | root (ChillerType attribute), 10  |
| 56  | root (DetectorType attribute), 14   |
|   | root (EquipmentIdentificationType attribute), 54                                      |
| IN  | root (ExperimentSetupType attribute), 47  |
| Nada Casteria ( - 1   | root (ExperimentStatisticType attribute), 48  |
| Node To Co (classic association and dellars) 50                   | root (ExperimentType attribute), 46   |
|   | root (FlowChemistryType attribute), 16  |
| No. 1 2 62 - 1 2 - 1 2 - 1 7 - 1 2 - 1                            | root (IdentificationType attribute), 52   |
| 11 4.6 4.   | root (IONType attribute), 18  |
| 57  | root (ManualControlType attribute), 20  |
|   | root (MFCTvne attribute) 23   |

| root (MonitoringType attribute), 56             | rseriesopc.model.experiment.experiment                             |
|---|--|
| root (NotificationType attribute), 57           | module, 45   |
| root (PumpType attribute), 25                   | rseriesopc.model.experiment.experimentSetup                        |
| root (R2Type attribute), 28                     | module, 47   |
| root ( <i>R4Type attribute</i> ), 31            | <pre>rseriesopc.model.experiment.experimentStatistics</pre>        |
| root (ReactionManagerType attribute), 48        | module, 48   |
| root (ReactorType attribute), 31                | rseriesopc.model.experiment.reactionManager                        |
| root (SF10Type attribute), 34                   | module, 48   |
| root (SoftwareIdentificationType attribute), 55 | rseriesopc.model.factories   |
| root (SyringePumpType attribute), 40            | module, 58   |
| root (UV150Type attribute), 42                  | <pre>rseriesopc.model.identification.equipmentIdentification</pre> |
| root (VBFRType attribute), 44                   | module, 54   |
| rSeries (RSeriesClient attribute), 3            | rseriesopc.model.identification.identification                     |
| RSeriesClient (class in rseriesopc.client), 3   | module, 52   |
| rseriesopc.client                               | rseriesopc.model.identification.softwareIdentification             |
| module, 3                                       | module, 55   |
| rseriesopc.model.base                           | rseriesopc.model.monitoring.monitoring                             |
| module, 58                                      | module, 56   |
| rseriesopc.model.devices.autosampler            | rseriesopc.model.notification.notification                         |
| module, 4                                       | module, 57   |
| rseriesopc.model.devices.azura                  | rseriesopc.model.reaction  |
| module, 8                                       | module, 58   |
| rseriesopc.model.devices.chiller                | rseriesopc.utils   |
| module, 10                                      | module, 59   |
| rseriesopc.model.devices.detector               | rSeriesStaus (MonitoringType attribute), 56                        |
| module, 14                                      | 13c11c35 caab (nonnoring type annionie), 50                        |
| rseriesopc.model.devices.factories              | S  |
| module, 16                                      | _  |
| rseriesopc.model.devices.flowChemistry          | sensorType (ChillerType attribute), 11                             |
| module, 16                                      | setBPRRegulationPressure() (SF10Type method),                      |
| rseriesopc.model.devices.ion                    | 38   |
| module, 18                                      | setChillerType() (ChillerType method), 13                          |
| rseriesopc.model.devices.manualControl          | setCollectionValveState() (AutoSamplerType                         |
| module, 20                                      | method), 7   |
| rseriesopc.model.devices.mfc                    | setCurrent() (IONType method), 19                                  |
| module, 23                                      | setFlowRate() (AzuraPumpType method), 9                            |
| rseriesopc.model.devices.module                 | setFlowRate() (MFCType method), 25                                 |
| module, 25                                      | setFlowRate() (PumpType method), 27                                |
| rseriesopc.model.devices.pump                   | setFlowRate() (SF10Type method), 38                                |
| module, 25                                      | setFlowRate() (SyringePumpType method), 42                         |
| rseriesopc.model.devices.r2                     | setGasFlowRate() (SF10Type method), 38                             |
| module, 28                                      | setILValveState() (PumpType method), 27                            |
|   | setLampPower() (UV150Type method), 43                              |
| rseriesopc.model.devices.r4                     | setLeaveRunning() (ChillerType method), 13                         |
| module, 31                                      | <pre>setMaxPressureLimit() (AzuraPumpType method),</pre>           |
| rseriesopc.model.devices.reactor                | 10   |
| module, 31                                      | setMaxPressureLimit() (R2Type method), 30                          |
| rseriesopc.model.devices.sf10                   | setMaxPressureLimit() (SF10Type method), 38                        |
| module, 34                                      | setOperationMode() (SF10Type method), 39                           |
| rseriesopc.model.devices.syringe                | setOscillation() (SF10Type method), 39                             |
| module, 40                                      | <pre>setPosition() (SyringePumpType method), 42</pre>              |
| rseriesopc.model.devices.uv150                  | setPressureLimit() (R2Type method), 31                             |
| module, 42                                      | setPressureLimit() (SF10Type method), 39                           |
| rseriesopc.model.devices.vbfr                   | <pre>setReactionPosition() (ReactionManagerType</pre>              |
| module, 44                                      | method), 52  |

```
setSensorType() (ChillerType method), 14
                                                      VBFRType (class in rseriesopc.model.devices.vbfr), 44
setSRValveState() (PumpType method), 27
                                                      voltage (IONType attribute), 18
setSRValveState() (SF10Type method), 39
                                                      volume (VBFRType attribute), 44
setTemperature (ChillerType attribute), 10
setTemperature() (ChillerType method), 14
setTemperature() (ReactorType method), 34
                                                      Wavelengths (DetectorType attribute), 14
setValveState() (SyringePumpType method), 42
                                                      wcValveState (ManualControlType attribute), 20
setVoltage() (IONType method), 19
setWCValveState() (ManualControlType method), 22
SF10Type (class in rseriesopc.model.devices.sf10), 34
softwareDeveloper
                          (SoftwareIdentificationType
         attribute), 55
                          (SoftwareIdentificationType
softwareDeveloper()
        method), 55
softwareIdentification
                             (IdentificationType
                                                 at-
         tribute), 52
SoftwareIdentificationType
                                     (class
                                                  in
         rseriesopc.model.identification.softwareIdentification),
softwareIdentifier (SoftwareIdentificationType at-
         tribute), 55
softwareIdentifier()
                          (SoftwareIdentificationType
        method), 55
softwareRevision (SoftwareIdentificationType at-
        tribute), 55
softwareVersion()
                          (SoftwareIdentificationType
        method), 56
srValveState (PumpType attribute), 25
srValveState (SF10Type attribute), 35
start() (IONType method), 19
startManualControl() (ManualControlType method),
startReaction() (ReactionManagerType method), 52
statistics (ExperimentType attribute), 46
stop() (IONType method), 19
stopAll() (ManualControlType method), 23
stopReaction() (ReactionManagerType method), 52
stopRoutines() (AutoSamplerType method), 7
switchValveToCollect() (AutoSamplerType method),
SyringePumpType
                               (class
                                                  in
         rseriesopc.model.devices.syringe), 40
Т
temperature (ChillerType attribute), 10
temperature (ReactorType attribute), 32
toLetter() (in module rseriesopc.utils), 59
U
uv150 (ManualControlType attribute), 20
UV150Type (class in rseriesopc.model.devices.uv150), 42
V
```

68 Index

valveState (SyringePumpType attribute), 40