APPLICATION NOTE
February 2024

securecell

ADVANCED BIOPROCESS MONITORING AND CONTROL VIA THE LUCULLUS® REST API INTERFACE

Written by **Stefan Hauer**, ZHAW (hatr@zhaw.ch) and
Pascal Vonlanthen, Securecell AG (pascal.vonlanthen@securecell.ch)

Lucullus®, Securecell's process information management system, allows for the automation of complex bioprocess operations and online calculations of performance metrics. Some advanced mathematical analyses surpass the capabilities of Lucullus®. In such cases, users require a simple and robust interface, which allows for communication between Lucullus® and the applications that fulfill their more specific needs for advanced data handling. The Lucullus® REST API interface meets these requirements, offering a seamless data exchange capability within a local network or over the internet. In this application note, we demonstrate how users can use the Lucullus® REST API interface for their applications to automate data analysis and process control tasks. Even though the given examples are all implemented in Python, they are easily transferrable to other popular coding languages such as MATLAB, C, or R.

The importance of interconnectivity

The transformation of biopharmaceutical manufacturing towards Bioprocessing 4.0 production concepts is challenging and involves integrating intensified, continuous, predictive, and autonomous operations. Key elements of Bioprocessing 4.0 are process automation, digitalization, and interconnection, leaving behind paper-based procedures, data silos, manual process control, and equipment, as well as software that cannot communicate with each other. In the last years, more and more processes and workflows in bioprocessing environments have been automated, digitalized, and interconnected. Nonetheless, end-to-end integration of processes and workflows is still the exception. The establishment of integrated platforms depends on the commitment of managers to invest in IT infrastructure and the acceptance of users to adapt to new technologies.

The Lucullus® process information management system is a software platform that integrates devices and enables data exchange across devices, laboratories, and enterprises, and therefore, serves as the glue among all the applications (Figure 1). Lucullus® supports all aspects of Bioprocessing 4.0 production concepts, enabling process automation, digitalization, and interconnection. Process data can be seamlessly transferred from the Lucullus® database to electronic laboratory notebooks (ELNs), manufacturing execution systems (MESs), or advanced data analysis tools (Phyton, MATLAB, R, DataHowLab, to mention just a few) by leveraging standardized communication protocols such as OPC or REST interfaces.

In this application note, we present a specific example of an interconnected bioprocessing ecosystem at the Zurich University of Applied Sciences (ZHAW) in Wädenswil. The interconnection of bioreactor controllers, in-line, on-line, and at-line analytical devices, and software leveraging the Lucullus® REST API interface enables uninterrupted and automated data acquisition and analysis. The described bioprocess at ZHAW is characterized by complete information transparency.

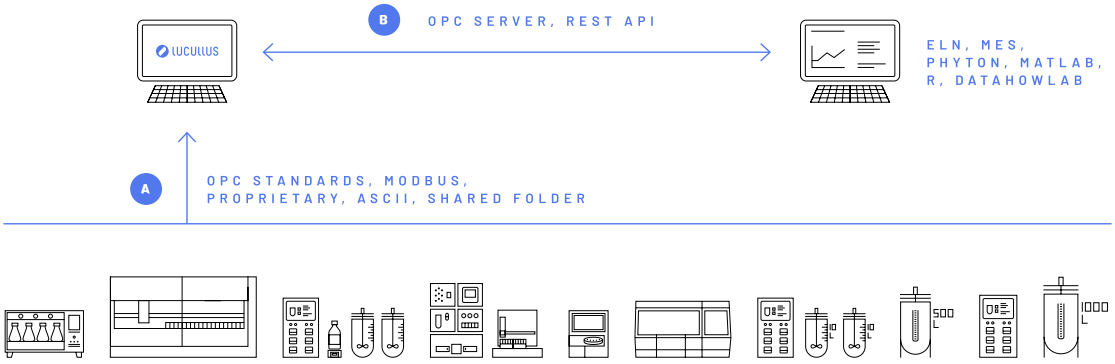

Figure 1: Lucullus® integrates more than 100 different devices and device models found in typical bioprocessing environments. (A) The devices come in a vast variety of versions, with different proprietary interfaces, and sometimes mismatching or old-fashioned technology which makes their integration a challenging task. Their integration into Lucullus® is realized by device-specific driver development or integration via so-called standard interfaces. (B) Through OPC or REST interfaces, Lucullus® also enables the transfer of the harmonized and structured data from the Lucullus® database to the respective customer-specific software solutions.

Base IP + Resource + Attributes
http:// 192.1.1.1:8080/lpims/rest/v1/reactors

Figure 2: REST API request consisting of Base IP + Resource prompting all bioreactors integrated in the Lucullus® process information management software (LPIMS).

For instance, to retrieve a list of all reactors with actively running processes, one would create a URL with the resource "reactors" and specify the attribute "running" as "true". A simple way to send such requests would be through the web browser, assuming the user is on the same network as the Lucullus® server (Figure 3).

IP address = http:// 192.1.1.1:8080/lpims/rest/v1/reactors?running=true

Figure 3: REST API request consisting of Base IP + Resource + Attributes prompting all running bioreactors integrated in the Lucullus® process information management software (LPIMS).

HTTP/HTTPS requests can be done just as easily with the "requests" library in Python. The following code snippet shows how this could be implemented in just a few lines. After retrieving the information for all reactors with actively running processes, the data can be further processed, for instance, to retrieve the process names:

```
import requests

auth = ("user", "password")
response = requests.get(
    "http://192.1.1.1:8080/lpims/rest/v1/reactors?running=true",
    auth=auth
)
process_names = [
    x["process"]["name"]
    for x in response.json()["data"]
]
process_ids = [
    x["process"][ "id"]
    for x in response.json()["data"]
]
print(process_names)
print(process_ids)
```

Code 1: Code snippet with the "requests" library in Python retrieving the process name and process ID of all running bioreactors integrated in the Lucullus® process information management software (LPIMS).

Another common task might be to retrieve the current port values during a running process. In the following code example, this is achieved in two steps:

1. Retrieve the identifier of the ports based on their name
2. Retrieve the current values of those ports from the specified reactor

```python
import requests

port_names = ["pO2", "Temperature"]
port_ids = []

for p in port_names:
    response = requests.get(
        ("http://192.1.1.1:8080/lpims/rest/v1/"
            + "ports?name={}".format(p)),
        auth=auth
    )
    id = response.json()["data"][0]["id"]
    port_ids.append(id)

link = (
    "http://192.1.1.1:8080/lpims/rest/v1/reactors/"
    + reactor_name
    + "?currentValues="
    + ",".join(port_ids),
)
response = requests.get(
    link, auth=auth
)
current_values = response.json()["data"]["process"]["currentValues"]
print(current_values)
```

Code 2: Code snippet with the "requests" library in Python retrieving the current port values of bioreactors integrated in the Lucullus® process information management software (LPIMS).

Finally, users would like to not only retrieve process data but also change setpoints during a running process. Note that this is only possible for ports that are set to "Output" in the System Administration tool of Lucullus®, are logged by default, and belong either to a hardware or logical device.

```python
link = (
    "http://192.1.1.1:8080/lpims/rest/v1/"
    + "signals?"
    + "portId={}".format(process_ID)
    + ",processId={}".format(port_ID)
)
response = requests.get(
    link,
    auth=auth
)
signal_id = response.json()["data"][0]["id"]
signal_id = response.json()["data"][0][ "id"]
```

Code 3: Code snippet with the "requests" library in Python retrieving the signal ID of a port where the value will be changed.

After having successfully retrieved the signal ID of the port to be changed, the "put" command of the "requests" library can be used to set the current value to 300:

```
link = "http://191.1.1.1:8080/lpims/rest/v1/signals/{}".format(signal_id)
headers={"Content-Type": "application/json"}
response = requests.put(
  link,
  data=json.dumps({"currentValue": 300}),
  auth=auth,
  headers=headers
)
```

Code 4: Code snippet with the "requests" library in Python changing the port value of the port with the retrieved signal ID to 300.

Process control via the REST API

Although the provided code examples are exclusively written in Python, data could be exchanged between Lucullus® and any software of preference as long as the software supports the transmission of HTTP/HTTPS requests. In the following section, we will show how to use data transfer based on the Lucullus® REST API for process control.

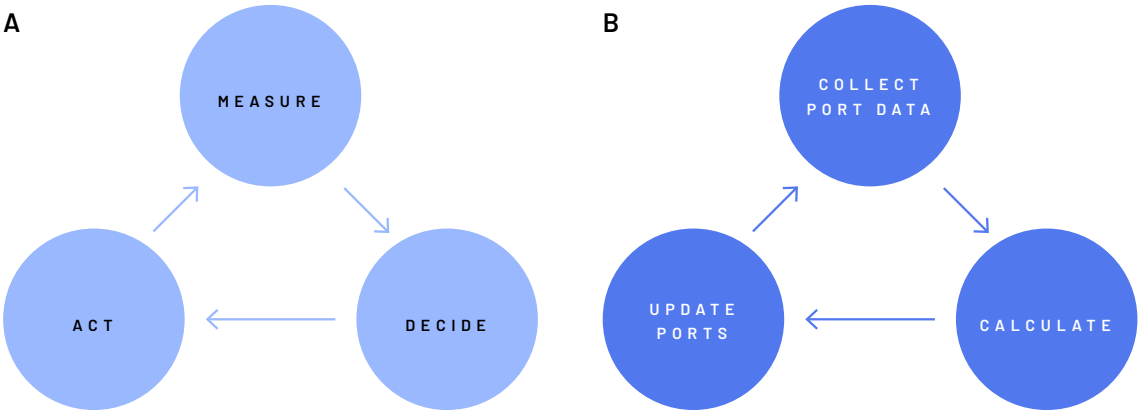

Figure 4: Advanced process monitoring and control principle. (A) The general process control principle includes the following points that are sequentially repeated: Measuring of CPPs, deciding based on process knowledge on the CPPs adjustments, and acting by adjusting respective CPPs. (B) The general process control principle is the framework for a general structure to send and retrieve data between Lucullus® and a programmed code: Collect port data, perform calculations to analyze the process state, and update ports based on collected/calculated data.

Any form of process control follows a general process control principle (Figure 4A):

1. Measurement: Critical process parameters (CPPs) are measured either in-line, on-line or at-line
2. Decision: Based on current and historical process data and expert knowledge, a decision to modulate a CCP is taken
3. Action: The CPP is adjusted. Afterward, the cycle of measurement, decision, and action starts again

A similar guiding principle can help to develop a general structure to send and retrieve data between Lucullus® and a programmed code. In a specified interval, the control algorithm might perform the following steps (Figure 4B):

1. Data retrieval: Retrieve the data from the ports of interest, either as the current values or the complete time series, based on the requirements of the respective application
2. Calculation: Based on the retrieved data, features of interest are calculated, e.g., specific and volumetric rates or estimated metabolite concentrations
3. Port update: Based on collected or calculated data, Lucullus® ports are updated, e.g., pH control setpoints, aeration rate, or pump settings

Researchers at Zürich University of Applied Sciences decided on a functional programming approach, which means that the calculation and update of ports are both based on functions. In contrast to an object-oriented paradigm, where code is structured into classes, this approach allows for easier testing as outputs are only dependent on inputs and not also on class attributes and states. Additionally, this approach is more straightforward for users who are not experienced with coding. In the following pseudocode example, we show how to implement a feedback loop with dedicated functions for data collection and update of the ports:

```
while True:
  collected_data = collect_port_data(process, port_names, auth)
  calculated_data = update_calculations(collected_data, calculated_data)
  ports_to_update = update_ports(collected_data, calculated_data, auth)
  update_ports(process, ports_to_update, auth)
  sleep.wait(60)
```

Code 5: Code snippet showing how to implement a dedicated function for data collection and port update.

As many data science routines in Python are built around pandas DataFrames, the "collect_port_data" function returns a DataFrame of the complete time series data. The used code is provided on GitHub via the following link. Users should consider encapsulating frequently used requests into easier-to-use functions or classes. This approach could enhance the efficiency and maintainability of the codebase.

Results

The Lucullus® REST API and Python were used to automate an industrial scale-down model of a lipid production process using a yeast strain. One challenging aspect of this production process is the yeast's metabolism which adapts to the changing process conditions. As a consequence, the specific substrate uptake rate may fluctuate. Therefore, traditional feeding techniques might lead to overfeeding when the feeding rate is not adjusted to the current biomass concentration and maximum substrate uptake rate and can lead to reduced productivity or even failed batches (Reichelt, et al., 2017). A simple and often applied solution to this problem is a pulsed feeding strategy: After the yeasts have consumed the available sugar, the feed solution is pulsed into the reactor to increase substrate levels. The pulsed feed strategy is often based solely on dissolved oxygen measurements (Paddon, 2013; Poontawee, 2020; Carsanba, 2021) but other process signals such as O_2 or CO_2 in the off-gas can also be used. This cycle of subsequent pulsing and waiting for substrate depletion continues until the end of the process. Therefore, there is no uncontrolled accumulation of the substrate inside the bioreactor.

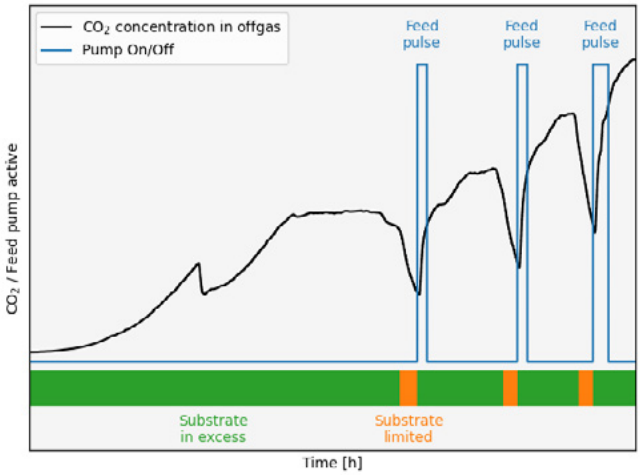

Figure 5: The black line depicts the CO_2 measurements in the off-gas (outliers removed and smoothed), while the blue line denotes the state of the feed pump over time. The green and orange bars beneath the process signals show the evaluation of the process state by the algorithm: While green means that substrate is still available in excess in the media, orange means that the substrate is limited. The curves depict a typical process over time. After the initial batch phase, the first substrate limitation is reached, clearly visible by a drop in the CO_2 signal. Based on this indicator among several, the feed pump is turned on to increase the substrate concentration up to a predefined value. Afterward, the cycle of waiting and pulsing starts again.

Figure 5 depicts a typical production process controlled via the pulse-feed strategy. Once every minute, thirty different measurement values are retrieved, such as pO_2, O_2, and CO_2. Afterward, the variables of interest are calculated such as oxygen uptake rate (OUR) and carbon dioxide evolution rate (CER) from O_2 and CO_2 measurements in the off-gas. When the oxygen uptake rate drops, an algorithm interprets this as "substrate limitation" and pulses feed solution. This approach efficiently minimizes the duration of substrate limitation.

To keep feed pulses reproducible, the substrate level is always replenished to the same concentration after each feeding cycle. Consequently, the estimation of the reactor volume is important. The reactor volume is estimated based on reactor balance values, feed balance values, and humidity in the off-gas. By combining these parameters, changing reactor volumes based on feeding, acid and base addition, sampling, and evaporation can be accounted for. Additionally, redundancy is used to correct unforeseen events, such as items being put on a balance by accident.

Finally, based on the off-gas measurements of O_2 and CO_2 and the time passing from feeding until substrate depletion, volumetric substrate uptake rates are estimated online. These parameters give operators additional information, allowing them to approximate both substrate concentrations and the expected depletion of the latest substrate pulse.

Conclusion

By using the Lucullus® process information management system together with Python, a complex yeast fermentation process for lipid production was fully automated. Advanced mathematical analyses such as machine-learning-based event detection and soft sensing were performed in Phyton. The process data collected in the overarching software Lucullus® was seamlessly transferred to Phyton via the Lucullus® REST API interface.

As a consequence, operation time was reduced while at the same time, reproducibility was increased, and process adaptation to a complex yeast metabolism was possible. In general, the use of the REST API expands the capabilities of Lucullus® to all state-of-the-art machine learning algorithms available and allows for easy online integration of already established data analysis routines.

Key results

With the functionality of the REST API, customers can integrate their data analysis and control workflows seamlessly with Lucullus®. This saves time and effort by working with the tools operators already know best

The power of advanced mathematical methods such as Kalman filters, metabolic flux analysis, or machine learning can be leveraged online and help make better-informed decisions

References

- (2020, May 8). Retrieved from RedHat: https://www.redhat.com/en/topics/api/what-is-a-rest-api
- Carsanba, E. P. (2021). Fermentation strategies for production of pharmaceutical terpenoids in engineered yeast. *Pharmaceuticals*.
- Gupta, L. (2023, December 1). Retrieved from RESTful API: https://restfulapi.net/
- Paddon, C. J. (2013). High-level semi-synthetic production of the potent antimalarial artemisinin. *Nature*.
- Poontawee, R. &. (2020). Feeding strategies of two-stage fed-batch cultivation processes for microbial lipid production from sugarcane top hydrolysate and crude glycerol by the oleaginous red yeast Rhodosporidiobolus fluvialis. *Microorganisms*.
- Reichelt, W. N., Brillmann, M., Thurrold, P., Keil, P., Fricke, J., & & Herwig, C. (2017). Physiological capacities decline during induced bioprocesses leading to substrate accumulation. *Biotechnology journal*.

SECURECELL AG

In der Luberzen 29
CH – 8902 Urdorf
+41 44 732 91 00
contact@securecell.ch