Updated on: September 08, 2021

Integrating Emerald Cloud Lab with your existing data systems

Emerald Cloud Lab (ECL) includes a sophisticated and powerful stand alone data ontology for analyzing and managing the data produced in the cloud lab. However, many users find it necessary to integrate ECL's systems with existing data systems like Data Lakes, Laboratory Information Management Systems (LIMS), or Electronic Laboratory Notebook (ELN) systems in order to:

1. Connect experiments performed in the cloud with those performed in other environments
2. Make it easy to ship samples from existing facilities to the cloud lab
3. Meet regulatory requirements about centralized data retention

ECL's API first approach to running and managing experiments makes this easy. This document outlines common strategies for getting such integrations up quickly and reliably.

Determining what data is needed

ECL likely produces considerably more data than other systems you may have worked with. This includes not only data around experiments, samples, and instruments, but metadata about the experiments, including lab conditions (e.g., temperature and humidity), performance and timing measurements for each step of the protocol, and results of troubleshooting or corrective interventions when required. The first step in any integration project is to select which of this data is needed.

Once the relevant data is selected, it is typically necessary to determine a mapping from how the data is stored in one source to how it is stored in the other. For example, the chemical composition of a solvent mixed into a reagent at one step in a protocol may be called two very different things in two different systems.

The complexity of building this mapping depends heavily on the existing data system. If the existing system is a structureless data lake, it is extremely simple to convert objects in ECL into JSON and dump them directly into the data lake. However, if the existing system is highly structured and opinionated on names of fields and structures of objects, it may take more work to build the mapping. The best way to build out this mapping is to directly compare some objects in ECL and your existing system and start to build out a mapping for each piece of relevant data. ECL can help with this if needed.

Selecting a Technical Strategy

Once a mapping between how data is stored in ECL and in your existing system, the next step is to select a technical strategy for ensuring that data stays in sync between the two systems. There are two main approaches to implementing the integration, and which is best depends on many factors, including:

1. The goals and requirements of the integration
2. Any security limitations that limit access to your Data Lake, LIMS, or ELN
3. Whether the integration will be owned by your scientific or IT teams

The two main approaches, with their pros and cons, are described below.

Using Command Center

The first option is to implement the integration in ECL's Command Center application. In this approach, Mathematica code is written to make API calls to your Data Lake, LIMS, or ELN system to export or import the data as needed. This Mathematica code is then triggered on ECL's systems manually by users, on a fixed schedule, or whenever relevant new data is created or needed depending on requirements.

The pros of this approach are:

- The integration happens naturally as users make use of ECL
- Users control the integration and can make changes to relevant data, data mapping, and frequency/triggers of integration as needed
- This approach is generally faster and simpler to implement

The cons of this approach are:

- Automatic schedule or event based triggering of the integration will only work if ECL's servers are able to access the Data Lake, LIMS or ELN
- You must work with your IT team to get information around how to access your <figure this out>

The details of this approach are described later in this document.

Using the Constellation API

The second option is to use the constellation API directly. The constellation API, which is the same API that powers all access to data via Command Center, allows direct downloading, searching, and uploading of data to ECL. In this way, it is possible to build a custom application in any desired language or framework that integrates ECL with your Data Lake, LIMS, or ELN.

The pros of this approach are:

- IT teams may use whatever technology or framework they prefer to implement the integration
- The integration may run on the same network as the Data Lake, LIMS, or ELN servers, which may make certain security practices easier to implement

The cons of this approach are:

- The control of this integration is not in the hands of the users and so making changes to the relevant data, mapping or triggering events may be harder
- You can't use any of the custom SLL functions
- It is often considerably more time consuming to build a custom integration

The details of this approach are described in later sections.

Using Command Center to interact with your Data Lake, LIMS, or ELN

Mathematica is able to perform any network requests required to call out to APIs of your Data Lake, LIMS, or ELN directly from Command Center. While every system is going to be slightly different, there is often a common pattern where there is a root resource for objects, which then accepts GET requests to download objects and POST requests to create objects. A simple example of this is:

```
(* Set the URL for your data store's API *)
LIMSUrl = "https://myhost:myport/path_to_api/";

(* Download some data from the data store *)
URLExecute[LIMSUrl <> "samples?queryid=mysample"]

(* Find some data on the ECL side - this pulls the model, name, and ID of
samples created in the last week *)
sampleData = Download[Search[Object[Sample, DateCreated > Now - 1 week && Model
!= Null, MaxResults -> 10], {ModelName: Model[Name], SampleName: Name,
SampleID: ID}]

(* Turn the data you downloaded from ECL into JSON for your data store *)
jsonData = ExportString[sampleData, "RawJSON"];

(* Upload the data to the data store *)
URLExecute[HTTPRequest[LIMSUrl <> "samples", <|"Method"->"POST",
"ContentType"->"application/json", "Body->jsonData|>]]
```

Note that the above will require a few tweaks to get working on your system: namely the URLs must be updated and the expected form of the jsonData may be different. ECL can help with this based off of your data store's documentation.

Once you have the basic integration working, the next steps are to implement the field mapping between ECL and the data store and to schedule the integration to run repeatedly.

Using the Constellation API

The constellation API provides the same functionality to access and manage data as does Command Center. The API is fully documented at

www.emeraldcloudlab.com/internal-developers-api

and SDK's are available at:

https://github.com/emeraldsci/constellation-sdk

Note that you will need to be invited to view the sdk, so please let us know if you would like access.