Proposal for a simple API  
for the purposes of University of Arizona INSPIRE

[API endpoints](#_Toc82667523)

[Turning a component on for a given lab](#_Toc82667524)

[Turning a component off for a given lab](#_Toc82667525)

[Querying the status of the component](#_Toc82667526)

[Authentication](#_Toc82667527)

[Configuration](#_Toc82667528)

[Interlock](#_Toc82667529)

[Interlock channels](#_Toc82667530)

[Networking](#_Toc82667531)

# API endpoints

Turning a component *on* for a given lab

When a customer session is started in iLab, we will record the **customer’s identity** and the **time the session h**as started as part of the reservation. At that point we tell the INSPIRE system to enable the corresponding component for a given lab:  
  
https://<hostname>/on?component=<component\_id>&lab=<lab\_id>

**Parameters:**

**component**: a unique identifier that tells the system which component needs to be turned on  
**lab:** a unique identifier that tells the system which lab the output photon stream should be sent to

The identifiers can be a number or an alphanumeric string, as long as they are unique.

Turning a component *off* for a given lab

When the researcher is done with their work, they can go to iLab and stop their session which will result in iLab sending a request to the following endpoint to turn off the photon stream:

https://<hostname>/off?component=<component\_id>&lab=<lab\_id>

The parameters are the same as above.

Querying the status of the component

To ensure that the *on* or *off* operation has succeeded and to be able to display the correct status in iLab, we need to be able to query whether the INSPIRE system considers a given component *on* or *off*:

https://<hostname>/query?component=<component\_id>&lab=<lab\_id>

# Authentication

To keep things simple and secure we can implement authentication based on an API key. As long as the communication with the target system is over HTTPS this key should stay secure from interception. It can also be changed periodically to enhance security. It will be entered in the iLab interface and then send as a header with the request:

X-API-Key: hOc2wGhPPraHMPxB4UYP6MXIyTgDoFTGxUnF/Wqn8lzVWIN2kljgQg==

The API key can be generated randomly, for example with this command, in a UNIX-like system:

openssl rand -base64 40

# Configuration

In iLab the needed configuration is stored in an Interlock object, which can have multiple channels. Each channel can be associated with a specific calendar, which can have rules related to training, authorization, billing etc. A calendar can also have multiple instances, to indicate that there are several identical resources that can be scheduled in parallel. Instances can be controlled indepdendently by the interlock system.

## Interlock

In the specific case of the API endpoints, you will need to create an “interlock” device, where you will have to enter the following information:

* **hostname:** the IP address or FQDN of the target machine where the API will be running
* **API key:** The key that will be sent to the API endpoint for authentication

## Interlock channels

For each possible combination of component and lab that will need to be scheduled and ran together, we need to configure a channel, which will have the following properties:

* **component:** The unique component identifier to be used in the API calls outlined above
* **lab:** The unique lab identifier to use in API calls

# Networking

The iLab Interlock module does not require for the API endpoints of the interlock devices to be exposed to the Internet at large. This is the purpose of the iLab Bridge, a small device that lives on the customer’s campus and will be the effective originator of the API calls. Therefore, the bridge should be able to reach the target machine where the API is running on port 443, or HTTPS.