iLab API User Manual

Table of Contents

Overview

The iLab Solutions API provides secure access to core resources, services, equipment, service requests, and billing data. It is designed to allow client applications to integrate with the iLab platform, automate workflows, and enhance reporting capabilities.

The API supports the following key workflows:

- Retrieve core, service, and equipment data
- Submit and update service requests
- Retrieve and add charges to service requests
- Access and manage custom forms and attachments
- Track milestones and project lifecycle stages

Getting Started

Authentication

The iLab API uses OAuth 2.0 Bearer Tokens for authentication. A valid token must be included in the header of each request:

```
Authorization: Bearer {token}
```

Tokens are managed at the core level through the **API Clients** section under the core's Administration tab (if API access is enabled at your institution).

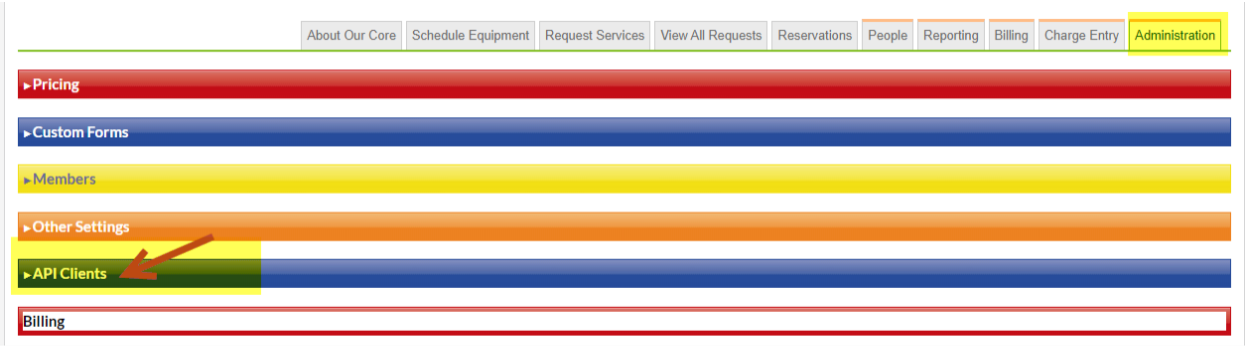
Obtaining a Client ID & Token

Client ID

A client ID serves to uniquely identify a client application to your core's API.

Token

Once a client ID has been generated, tokens can be associated with the client ID that provide a designated level of access to data through the API.



Please note that tokens, by default, expire one year from the date of initial generation. If you would like to set a longer expiration period for your token, take the following steps:

Go to the *API Clients* section within your core Administration tab. Click on the link that says "# token" to expand the details of the particular token that has expired. Place your cursor over the expiration date and a red pencil will appear. Click on the date and a calendar pop-up will appear that will allow you to set a new expiration date:

Using your client ID and token to request data through the API

The access token must be included in the header of each request to the API:

Authorization: bearer %token% \r\n

Each request to the API server must contain the access token. This will ensure that client applications have the appropriate access levels. This token must be kept secret on the client side, as it can represent a potential for a man-in-the-middle attack. Note, the iLab server will reject any requests that are not using SSL, but if you attempt plain-text access the token could theoretically captured by a third party.

Important: Always use HTTPS when accessing the API. Plain HTTP requests will be rejected.

API Endpoints

- **US (MY/General)**: https://api.ilabsolutions.com
- **EU (Europe)**: https://api-eu.ilabsolutions.com
- **AU (Australia)**: https://au-api.ilab.agilent.com

- **Stanford:** https://stanford-api.ilabsolutions.com

Requests & Responses

Rate Limiting

To ensure platform stability, API usage is rate limited to **300 requests per token per minute**. Exceeding this limit will result in a `429 Too Many Requests` response.

HTTP Methods

The iLab API is RESTful and supports the following HTTP methods:

- `GET`: Retrieve resources
- `POST`: Create new resources
- `PUT`: Update existing resources
- `DELETE`: DELETE resources

Response Format

Responses are returned in either **JSON** (default) or **XML** format, depending on the URL extension:

- `.json` → JSON response
- `.xml` → XML response

Example:

```
GET https://{domain}/v1/cores.json
```

Pagination

Pagination metadata is provided in the `ilab-metadata` object:

```
<ilab-metadata>
  <next-page nil="true"/>
  <offset>25</offset>
  <previous-page>1</previous-page>
  <total>48</total>
  <total-pages>1</total-pages>
</ilab-metadata>
```

Use the page parameter to retrieve additional pages: `GET https://{domain}/v1/resource?page=2`

Resources

Cores

Cores are the root resource of the API

- `GET /cores`: Lists all accessible cores
- `GET /cores/{id}`: Individual core details

Services

- `GET /cores/{id}/services`: List core services
- `GET /cores/{id}/services/{service_id}`: Service details
- `PUT /cores/{id}/services/{service_id}`: Update service

Prices

- `GET /cores/{id}/services/{service_id}/prices`
- `GET /cores/{id}/services/{service_id}/prices/{price_id}`
- `PUT /cores/{id}/services/{service_id}/prices/{price_id}`
- `DELETE /cores/{id}/services/{service_id}/prices/{price_id}`

Service Requests

List service requests

- `GET /cores/{id}/service_requests`:
- `GET /cores/{id}/service_requests/{request_id}`
- `POST /cores/{id}/service_requests`
- `PUT /cores/{id}/service_requests/{request_id}`

Example Service Request GET Response (XML):

```
<ilab-response>
  <service-request>
    <id>493769</id>
    <name>Protein Quantification - Batch 7</name>
    <description>Batch of 24 samples for protein quantification</description>
    <state>processing</state>
    <created-at type="datetime">2025-05-14T13:52:00Z</created-at>
    <completed-on type="datetime" nil="true"/>
    <start-on type="datetime">2025-05-15T09:00:00Z</start-on>
    <end-on type="datetime" nil="true"/>
    <has-recurring>false</has-recurring>
    <projected-cost>1200.00</projected-cost>
    <summary/>
    <owner>
      <id>30125</id>
      <email>researcher1@institution.edu</email>
      <first-name>Jane</first-name>
      <last-name>Doe</last-name>
    </owner>
    <pi>
      <id>4521</id>
```

```
      <email>pi_lab@institution.edu</email>
      <first-name>Dr. John</first-name>
      <last-name>Smith</last-name>
    </pi>
    <service-rows>
      <service-row>
        <id>789123</id>
        <position>1</position>
        <type>charge</type>
        <name>Protein Quantification (per sample)</name>
        <quantity>24</quantity>
      </service-row>
    </service-rows>
    <milestones type="array">
      <milestone>
        <id>21214</id>
        <name>Samples Received</name>
        <completed-on type="datetime">2025-05-15T10:15:00Z</completed-on>
      </milestone>
    </milestones>
  </service-request>
</ilab-response>
```

Request Filtering

Supported by parameters: `has_recurring`, `from_date`, `to_date`, `states(see below)`

Example:

GET *https://{domain}.com/v1/cores/1234/service_requests.xml?q=samplename&has_recurring=1&to_date=2013-03-12T12:54Z&states=cancelled,disagreement*

Filter name	Available values	Default values
`has_recurring`	0 or 1 (optional)	Both 0 and 1
`from_date`	string ISO 8601 formatted in UTC (optional)	Time.now - 2.years
`to_date`	string ISO 8601 formatted in UTC (optional)	Time.now + 1.day
`Q`	string for fulltext search (optional)	Not applied
`Name`	string for exact match search by name (optional)	Not applied
`Order`	order field name	created_at

`States`	valid request states(comma separated list): cancelled, completed, core_disagreement, disagreement, draft, equipment_scheduling, financials_approved, financials_rejected, needs_financial_reapproval, processing, proposed, requested, researcher_in_agreement, service_center_in_agreement	All states

Retrieve Service Requests

Parameter	Description	Default
`has_recurring`	0 or 1 (optional)	Both 0 and 1
`from_date`	ISO 8601 UTC date	2 years ago
`to_date`	ISO 8601 UTC date	Tomorrow
`Q`	Full-text search (optional)	Not applied
`Name`	Exact name match	Not applied
`Order`	Sort field (e.g. `created_at`)	`created_at`
`States`	Comma-separated list of request states	All states

Update Service Request

- `PUT /cores/{id}/service_requests/{request_id}`

In the body of the request, include the same XML you received when you retrieved the resource, but modify any fiel ds that you would like to change. For example, if you want to change the first price and the name of the service yo u would pass on:

Example

```
<service>
<description><p>They were on ice</p></description>
<name>Elaborate Fish Melting</name>
<prices type="array">
<price>
<id type="integer">65</id>
<price type="float">8.0</price>
<price-type>
<id type="integer">13</id>
<name>External</name>
</price-type>
<unit>
<abbreviation>ea</abbreviation>
<description>each</description>
<id type="integer">37</id>
</unit>
</price>
</prices>
</service>
```

Normally, you'd only pass along the attributes that you want to update. If you want to set an attribute to nil, you n eed to specify that in the XML/JSON explicitly, e.g.:

```
<category nil="true"/>
```

Public Visibility setting controls if service is visible on the core's landing page and searchable on the institution l anding page.

Allowed values of public_visibility allowed include:

- 0 not visible on landing page
- 1 visible on landing page

Updatable fields

name	Description	State	completed_on	start_on	end_on	quote_expires_on	has_recurring	projected_cost	summary

Create Service Request

- `POST /cores/{id}/service_requests`

  Example:

```
<service_request>
  <owner_email>existing_user@email.com</owner_email>
  <pi_email>optional_pi@email.com</pi_email> <!-- optional -->
  <name>Optional Request name</name> <!-- optional -->
  <state>processing</state> <!-- optional; default is 'completed' -->
</service_request>
```

Create Service Request Fields

Field	Required	Description
`owner_email`	Yes	Email of an iLab user; becomes request owner
`pi_email`	Optional	PI email (used if user belongs to multiple groups)
`Name`	Optional	Request name (if omitted, auto-generated)
`State`	Optional	Initial request state; allowed: `proposed`, `needs_financial_reapproval`, `processing`, `completed`

Service Request States & Transitions

- **Allowed ``** values when creating or updating requests via API:**
  - `proposed`, `needs_financial_reapproval`, `processing`, `completed`

Full state transitions (internal lifecycle)

Event	From States (examples)	To State
`Propose`	`Draft`	`proposed`
`skip_approval`	`Proposed`	`financials_approved`
`skip_financials`	`draft`, `proposed`, `service_center_in_agreement`, `researcher_in_agreement`	`financials_approved`
`researcher_agree`	`proposed`, `disagreement`, **etc.**	`researcher_in_agreement`

Event	From States (examples)	To State
service_center_agree	researcher_in_agreement, needs_additional_approval	service_center_in_agreement
request_item	service_center_in_agreement, proposed	requested
approve_financials	requested, proposed, service_center_in_agreement	financials_approved
Process	financials_approved	processing
Complete	processing, received	completed
request_reapproval	financials_approved, processing, completed	needs_financial_reapproval
Cancel	Many states	cancelled

Note: Other transitions (e.g. **equipment_scheduling**, **disagreement**, **needs_department_approval**) are system-controlled and may not be set via the API directly.

Important: If you attempt to set an invalid transition via API (example: **PUT** a completed state on a request still in **draft**), the API will return a **422 Unprocessable Entity** error

Valid request states:

- cancelled, completed, core_disagreement, disagreement, draft, equipment_scheduling, financials_approved, financials_rejected, needs_financial_reapproval, processing, proposed, requested, researcher_in_agreement, service_center_in_agreement

Validations & Error Responses

HTTP Code	Error Message
404	Owner not found
404	Owner does not have access to core or is not a Core Employee
404	Owner is not a member of any Group
404	PI not found
404	PI is not in owner's Groups

Allowed billing statuses:

cancelled	not_ready_to_bill	ready_to_bill	not_billable	pro_bono	billing_initialized	billed

Allowed work statuses:

`proposed`	`financials_approved`	`processing`	`completed`	`cancelled`

Service Request Rows

- `GET /cores/{id}/service_requests/{request_id}/service_rows`

Service Request Attachments

- `GET /attachments/{attachment_id}`
- `POST /attachments`
- `DELETE /attachments/{attachment_id}`

Required params for adding attachments:

  - `object_class=ServiceItem`
  - `id={service_request_id}`

Restricted file types: `.bat`, `.exe`, `.tar` and associated MIME types.

Custom Forms

Custom forms are used to collect information from customers that is required to complete a service request. A form may contain a combination of field types: user-entered data, selected services/charges, file uploads, and instructional content.

Custom forms are linked to service requests and accessed via:

- `GET /cores/{id}/service_requests/{request_id}/custom_forms`
- `GET /attachments/{attachment_id}` **(for files attached through the form)**

Custom Form Field Types

Field Type	Description
`String`	Text field
`text_section`	Text block (static content, not user-in**PUT**)
`Select`	Dropdown/select menu
`Radio`	Radio buttons
`Checkbox`	Checkbox

Field Type	Description
`Charges`	Charge field (links services to form)
`File`	File upload
`file_no_upload`	Static file reference / download link
`Help`	Static help text or instructions

Example GET Response (XML):

```
<custom-forms type="array">
  <custom-form>
    <id>29385</id>
    <name>Flow Cytometry Setup</name>
    <fields type="array">
      <field>
        <name>Sample Type</name>
        <type>select</type>
        <choices>,Whole Blood,PBMC,Lysate</choices>
        <value>PBMC</value>
        <required>true</required>
      </field>
      <field>
        <name>Upload Sample Spreadsheet</name>
        <type>file</type>
        <value>attachment_id_1231</value>
      </field>
      <field>
        <name>Requested Services</name>
        <type>charges</type>
        <value type="array">
          <value>217701</value>
        </value>
        <required-services type="array">
          <required-service>
            <id>217701</id>
            <name>Flow Cytometry - Per Sample</name>
            <quantity>24</quantity>
          </required-service>
        </required-services>
      </field>
    </fields>
  </custom-form>
</custom-forms>
```

Note: Updating custom forms via the API is not supported. Forms can only be retrieved.

Attachments linked to custom forms can be retrieved separately using their `attachment_id`.

Milestones

- `GET /cores/{id}/service_requests/{request_id}/milestones`

- PUT /cores/{id}/service_requests/{request_id}/milestones/{milestone_id}

Charges

- GET /cores/{id}/service_requests/{request_id}/charges
- POST /cores/{id}/service_requests/{request_id}/charges
- PUT /cores/{id}/service_requests/{request_id}/charges/{charge_id}

Equipment

- GET /cores/{id}/equipment

Error Handling

HTTP Status	Meaning
400	Bad Request (invalid inPUT, malformed syntax)
401	Unauthorized (invalid or expired token)
404	Not Found (invalid URL or missing resource)
422	Unprocessable Entity (validation error)
429	Too Many Requests (rate limit exceeded)
500	Internal Server Error

404 HTTP status Not Found API Errors Examples

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<hash>
  <error>Object not found</error>
  <message>The API server could not find the object you specified</message>
</hash>
```

JSON:

```
{"error":"Object not found","message":"The API server could not find the object you speci
fied"}
```

If you encounter persistent `500` errors, please contact **ilabsupport@agilent.com** with details about the request.