# Introspect Cloud API
# User guide

### Document Change History

| Date | Version | Change | Author |
|---|---|---|---|
| 2026-01-09 | 1.0 | Initial version | Simon Evieux |

TECAN.

## Table of Contents


**1 Introduction 4**
1.1 Purpose 4
1.2 Scope 4
1.3 Abbreviations 4
1.4 Base URL 4
1.5 Constraints 4

**2 Overview 4**

**3 Installation 4**

**4 Authentication 5**
4.1 Generation of the original token in Introspect 5
4.1.1 Schema 5
4.1.2 Example 5
4.2 Token refresh 6
4.2.1 Endpoint 6
4.2.2 Parameters 6
4.2.3 Example Request: 6
4.2.4 Response Schema 6
4.2.5 Resonse example 7
4.2.6 Error handling 7
4.3 Safety 7

**5 Operation - Cloud API for Usage Data 8**
5.1 Endpoints 8
5.1.1 Get Instruments 8
5.1.1.1 Endpoint 8
5.1.1.2 Schema 8
5.1.1.3 Response Example: 9
5.1.2 Get Instrument Usage (Historical) Data 10
5.1.2.1 Endpoint 10
5.1.2.2 Parameters 10
5.1.2.3 Validation Rules 10
5.1.2.4 Example Requests 10
5.1.2.5 Response schema 11
5.1.2.6 Response Example 13
5.1.3 Get Instrument Indicators 15
5.1.3.1 Endpoint 15
5.1.3.2 Parameters 15
5.1.3.3 Validation Rules 15
5.1.3.4 Example Request 15
5.1.3.5 Response Schema 16
5.1.3.6 Response Example 17
5.2 Error Handling 19
5.2.1 Error Response Example 19

**6 Usage Examples – Cloud API for Usage Data 20**
6.1 Complete Workflow 20
6.2 Using curl on Windows Command Line 20

**7 Operation - Live Data API 21**

7.1 Choice of instruments 21
7.2 Connection to the SignalR hub and handling of messages 22
7.2.1 Connection to the Hub 22
7.2.2 Automatic Reconnection 22
7.2.3 Messages 23
7.2.4 liveMessageReceived: 23
7.2.4.1 Schema 23
7.2.4.1.1 MessageType 24
7.2.4.1.2 MessageSeverity 25
7.2.4.2 Example of a liveMessageReceived 25
7.2.5 instrumentStateChanged 26
7.2.5.1 Schema 26
7.2.5.1.1 InstrumentState 27
7.2.5.2 Example of a instrumentStateChanged Message 27
7.3 Safety 27

**8 Usage Examples – Live Data API 28**
8.1 Complete Workflow 28
8.2 Short example of a console application 28

**9 Support 30**

**10 Appendix 30**

# 1 Introduction

## 1.1 Purpose

This document provides a user guide for accessing and using the Tecan Cloud API for instrument usage (historical) data and the Live API for instruments connected to Introspect.

## 1.2 Scope

The guide covers authentication, usage, endpoints, error handling, example requests for the Cloud API and an implementation example.

## 1.3 Abbreviations

API: Application Programming Interface
GUID: Globally Unique Identifier
ISO: International Organization for Standardization

## 1.4 Base URL

All API endpoints are prefixed with: https://cloud-api.tecan.com/

## 1.5 Constraints

Rate Limit: 6 calls per minute. Exceeding this limit returns HTTP 429. Wait for the next minute to continue.

# 2 Overview

The Cloud API for usage data provides endpoints to:
- Retrieve instrument metadata
- Access historical usage data
- Obtain instrument indicators

All endpoints require authentication (as described in 4) and are subject to rate limits.

The Live API allows connection to a SignalR hub and all messages published to this hub within Introspect are forwarded to any client connected to the Live API.

# 3 Installation

No installation is required for API usage.
Ensure you have network access.
For the Cloud API for usage data ensure to have a tool (such as curl or Postman) to make HTTP requests.
For the Live Data API ensure you have a tool to implement a connection to a SignalR hub.

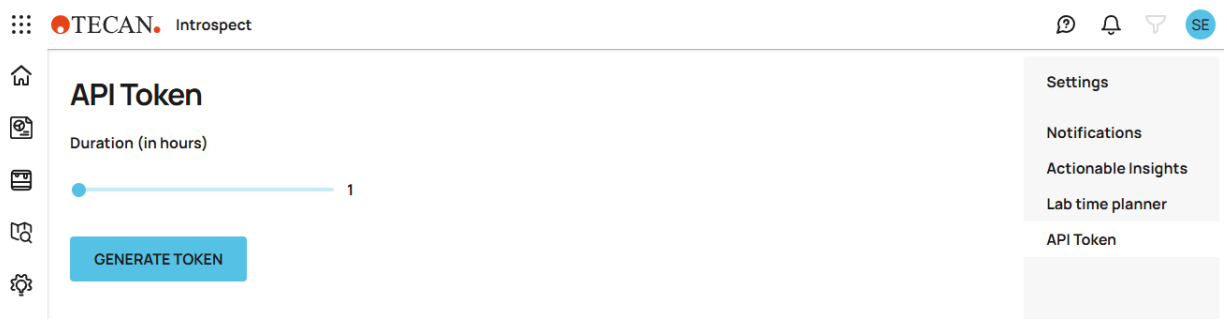
# 4 Authentication

All endpoints require an Authorization header with a valid token.

## 4.1 Generation of the original token in Introspect

The first token needs to be manually requested from the Introspect application. This can be done by users with the Lab Manager role.

1. Go to the Introspect API Token page
   https://digital.tecan.com/introspect/settings/api-token-settings




2. Select the Duration
3. Click Generate Token.
4. A JSON file containing your token: {"access_token": "TOKEN"} is downloaded in the configured download destination on your computer.

### 4.1.1 Schema

The file downloaded is a json file with the following schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "access_token": {
      "type": "string",
      "description": "JWT access token"
    }
  },
  "required": ["access_token"],
  "additionalProperties": false
}
```

### 4.1.2 Example

Here is an example of the file that is downloaded

```
{
    "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwczovL3RlY2FuLmNvbS9vcmdhbml6YXRpb25faWQiOiJhNGY2NmY3Ny1lYTg2LTQ3MjktODlmMy01MjZkYWY3ZDk5ZjEiLCJodHRwczovL3RlY2FuLmNvbS91c2VyX
}
```

For the Usage Data Cloud API, you will need to use the TOKEN value in the Authorization header for all API requests, including the refresh token.
The token is valid for the selected duration. Default is 1 hour, max is 168 hours (7 days)

For the Live Data, you will need to use the TOKEN value in the Authorization header for the hub connection. Once the connection is established, even if the token expires, the connection remains active.
If connection is lost and all reconnect trials have failed, a new valid token will be needed to re-establish the connection.

### 4.2 Token refresh

Before the end of the lifetime of a token, you can call a refresh token endpoint in order to get a new token to be used in your subsequent requests. If you fail to get a refresh token, a new token will need to be manually generated as described in 4.1

#### 4.2.1 Endpoint

GET /api/Tokens/refresh

#### 4.2.2 Parameters

durationInHours (optional, integer)

This is the validity in hours of the new token. Default is 1 hour, max is 168 hours (7 days) as when the token is requested from Introspect.

#### 4.2.3 Example Request:

GET /api/Tokens/refresh?durationInHours=2

#### 4.2.4 Response Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "access_token": {
      "type": "string",
      "description": "JWT access token"
    }
  },
  "required": ["access_token"],
  "additionalProperties": false
}
```

### 4.2.5 Resonse example


```
{
   "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwczovL3RlY2FuLmNvbS9vcmdhbml6YXRpb25faWQiOiJhNGY2NmY3Ny1lYTg2LTQ3MjktODlmMy01MjZkYWY3ZDk5ZjEiLCJodHRwczovL3RlY2FuLmNvbS91c2VyX
}
```


### 4.2.6 Error handling

Endpoint return appropriate HTTP status codes:
- 200 OK: Returns a new token
- 400 Bad Request: Invalid parameters or validation errors
- 401 Unauthorized: Authentication token missing or invalid
- 403 Forbidden: Insufficient permissions

## 4.3 Safety

No specific safety instructions are required for API usage. Ensure secure handling of authentication tokens.

# 5 Operation - Cloud API for Usage Data

## 5.1 Endpoints

### 5.1.1 Get Instruments

This endpoint returns the list of all instruments available for the user who has issued the original token in Introspect in json format.

#### 5.1.1.1 Endpoint

GET /api/Instruments/metadata

#### 5.1.1.2 Schema

The response on the Get instruments endpoint is as follows


```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "instrumentId": { "type": "string" },
      "type": { "type": "string" },
      "alias": { "type": "string" },
      "serialNumber": { "type": "string" },
      "size": { "type": "number" },
      "labId": { "type": "string" },
      "labDescription": { "type": "string" },
      "location": {
        "type": "object",
        "properties": {
          "country": { "type": "string" },
          "state": { "type": "string" },
          "city": { "type": "string" },
          "street": { "type": "string" },
          "streetNumber": { "type": "string" },
          "postalCode": { "type": "string" },
          "latitude": { "type": "number" },
          "longitude": { "type": "number" },
          "additionalInformation": { "type": ["string", "null"] }
        },
        "required": [
          "country",
          "state",
          "city",
          "street",
          "streetNumber",
          "postalCode",
          "latitude",
          "longitude",
          "additionalInformation"
```

```
      ]
    },
    "registrationDate": { "type": "string", "format": "date-time" },
    "lastConnectedTime": { "type": "string", "format": "date-time" }
  },
  "required": [
    "instrumentId",
    "type",
    "alias",
    "serialNumber",
    "size",
    "labId",
    "labDescription",
    "location",
    "registrationDate",
    "lastConnectedTime"
  ]
 }
}
```


### 5.1.1.3 Response Example:


```
[
 {
  "instrumentId": "Fluent~2207014657.780",
  "type": "Fluent",
  "alias": "Kurt",
  "serialNumber": "2207014657",
  "size": 780,
  "labId": "bb7b5c10-30b6-486a-aaab-3399ced013b0",
  "labDescription": "Service Cockpit account with Tecan instruments for demo purposes",
  "location": {
    "country": "Switzerland",
    "state": "Zurich",
    "city": "Hombrechtikon",
    "street": "Garstligweg",
    "streetNumber": "6",
    "postalCode": "8634",
    "latitude": 47.24608,
    "longitude": 8.78031,
    "additionalInformation": null
  },
  "registrationDate": "2023-02-22T10:28:44+00:00",
  "lastConnectedTime": "2026-01-31T18:27:07.367+01:00"
 }
]
```


The labId is the internal id of the lab in which the instrument is located (it will be needed when the user needs to retrieve the data for all the instruments in a given lab)

### 5.1.2 Get Instrument Usage (Historical) Data

This is the main endpoint that retrieves the usage data for specified instruments within a date/time range.

#### 5.1.2.1 Endpoint

GET /api/Instruments/usageData

#### 5.1.2.2 Parameters

FromDate (optional, ISO 8601): Start datetime (default: yesterday)
ToDate (optional, ISO 8601): End datetime (default: today)
LabIds (optional, array of GUID): Filter by lab ID
InstrumentIds (optional, array of string): Filter by instrument IDs
Page (optional, integer): Page number (default: 1)
PageSize (optional, integer): Items per page (default: 100, max: 1000)

If LabIds and InstrumentIds are skipped, no filter is applied on instruments / labs and data for all available instruments will be returned

#### 5.1.2.3 Validation Rules

LabIds must be valid GUIDs.
InstrumentIds must be non-empty strings.

#### 5.1.2.4 Example Requests

GET /api/Instruments/usageData?FromDate=2023-01-01&ToDate=2023-01-31&Page=1&PageSize=50

GET /api/Instruments/usageData?FromDate=2025-06-03T15:00:00+03:00&InstrumentIds=InstrumentId.1&InstrumentIds=InstrumentId.2

### 5.1.2.5 Response schema


```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "data": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "instrumentId": { "type": "string" },
          "organizationId": { "type": "string" },
          "completedRun": {
            "type": "object",
            "properties": {
              "id": { "type": "string" },
              "startTime": { "type": "string", "format": "date-time" },
              "endTime": { "type": "string", "format": "date-time" },
              "runName": { "type": "string" },
              "status": { "type": "string" },
              "failedSamples": { "type": "integer" },
              "flaggedSamples": { "type": "integer" },
              "skippedSamples": { "type": "integer" },
              "successfulSamples": { "type": "integer" },
              "methods": {
                "type": "array",
                "items": {
                  "type": "object",
                  "properties": {
                    "name": { "type": "string" },
                    "startTime": { "type": "string", "format": "date-time" },
                    "endTime": { "type": "string", "format": "date-time" }
                  },
                  "required": ["name", "startTime", "endTime"]
                }
              },
              "pauses": {
                "type": "array",
                "items": {
                  "type": "object",
                  "properties": {
                    "startTime": { "type": "string", "format": "date-time" },
                    "endTime": { "type": "string", "format": "date-time" }
                  },
                  "required": ["startTime", "endTime"]
                }
              },
              "errors": {
                "type": "array",
                "items": {}
```

```
    },
    "consumables": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "consumableName": { "type": "string" },
          "type": { "type": "string" },
          "logTime": { "type": "string", "format": "date-time" },
          "value": { "type": "number" }
        },
        "required": ["consumableName", "type", "logTime", "value"]
      }
    },
    "reagents": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "reagentName": { "type": "string" },
          "unit": { "type": "string" },
          "logTime": { "type": "string", "format": "date-time" },
          "value": { "type": "number" }
        },
        "required": ["reagentName", "unit", "logTime", "value"]
      }
    },
    "samples": {
      "type": "array",
      "items": {}
    },
    "genericVariables": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "dateTime": { "type": "string", "format": "date-time" },
          "introspectKey": { "type": "string" },
          "introspectValue": { "type": "string" }
        },
        "required": ["dateTime", "introspectKey", "introspectValue"]
      }
    }
  },
  "required": [
    "id",
    "startTime",
    "endTime",
    "runName",
    "status",
    "failedSamples",
```

```
            "flaggedSamples",
            "skippedSamples",
            "successfulSamples",
            "methods",
            "pauses",
            "errors",
            "consumables",
            "reagents",
            "samples",
            "genericVariables"
          ]
        }
      },
      "required": ["instrumentId", "organizationId", "completedRun"]
    }
  },
  "page": { "type": "integer" },
  "pageSize": { "type": "integer" }
 },
 "required": ["data", "page", "pageSize"]
}
```


### 5.1.2.6 Response Example


```
{
 "data": [
  {
   "instrumentId": "Fluent~2303005567.780",
   "organizationId": "b3fb083b-31ab-4274-8e16-63a120eec324",
   "completedRun": {
    "id": "8c0f3866-020c-4f22-80b7-2983f3083af6",
    "startTime": "2026-02-05T10:22:18.255+01:00",
    "endTime": "2026-02-05T13:22:23.538+01:00",
    "runName": "Method - PacBio_SRE_HiFi_MCA96_v3.0",
    "status": "CompletedSuccessfully",
    "failedSamples": 0,
    "flaggedSamples": 0,
    "skippedSamples": 0,
    "successfulSamples": 16,
    "methods": [
     {
      "name": "Introspect variables",
      "startTime": "2026-02-05T13:22:12.826+01:00",
      "endTime": "2026-02-05T13:22:12.92+01:00"
     },
     {
      "name": "Create report to confirm protocol complete",
      "startTime": "2026-02-05T13:22:12.925+01:00",
      "endTime": "2026-02-05T13:22:22.618+01:00"
     }
    ],
```

```
      "pauses": [
        {
          "startTime": "2026-02-05T13:19:34.898+01:00",
          "endTime": "2026-02-05T13:20:47.408+01:00"
        },
        {
          "startTime": "2026-02-05T13:21:43.649+01:00",
          "endTime": "2026-02-05T13:22:11.13+01:00"
        }
      ],
      "errors": [],
      "consumables": [
        {
          "consumableName": "Labware",
          "type": "96 Deep Well 450ul DPNGS",
          "logTime": "2026-02-05T13:22:23.513+01:00",
          "value": 1
        },
        {
          "consumableName": "Labware",
          "type": "96 Deep Well 2ml",
          "logTime": "2026-02-05T13:22:23.513+01:00",
          "value": 1
        }
      ],
      "reagents": [
        {
          "reagentName": "SRE",
          "unit": "Microliter",
          "logTime": "2026-02-05T13:22:23.52+01:00",
          "value": 1040
        }
      ],
      "samples": [],
      "genericVariables": [
        {
          "dateTime": "2026-02-05T13:22:12.906+01:00",
          "introspectKey": "NumberOf200ulMCADiTisUsed",
          "introspectValue": "64"
        }
      ]
    }
  }
 ],
 "page": 1,
 "pageSize": 15
}
```

### 5.1.3 Get Instrument Indicators

This endpoint retrieves indicators data for specified instruments within a date/time range.

#### 5.1.3.1 Endpoint

GET /api/Instruments/indicators

#### 5.1.3.2 Parameters

FromDate (optional, ISO 8601): Start datetime (default: yesterday)
ToDate (optional, ISO 8601): End datetime (default: today)
LabIds (optional, array of GUID): Filter by lab ID
InstrumentIds (optional, array of string): Filter by instrument IDs
Page (optional, integer): Page number (default: 1)
PageSize (optional, integer): Items per page (default: 100, max: 1000)

If LabIds and InstrumentIds are skipped, no filter is applied on instruments / labs and data for all available instruments will be returned

#### 5.1.3.3 Validation Rules

Date range cannot exceed 31 days.
LabIds must be valid GUIDs.
InstrumentIds must be non-empty strings.

#### 5.1.3.4 Example Request

GET /api/Instruments/indicators?LabIds=a50e8400-e29b-41d4-0716-446655440000&InstrumentIds=InstrumentId.1&InstrumentIds=InstrumentId.2&FromDate=2025-07-01&ToDate=2025-07-07

### 5.1.3.5 Response Schema


```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "data": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "instrumentId": { "type": "string" },
          "score": {
            "type": "object",
            "properties": {
              "current": { "type": "number" },
              "previous": { "type": "number" },
              "trendPercentage": { "type": "number" },
              "trendDirection": { "type": "string" }
            },
            "required": [
              "current",
              "previous",
              "trendPercentage",
              "trendDirection"
            ]
          },
          "utilization": {
            "type": "object",
            "properties": {
              "current": { "type": "number" },
              "previous": { "type": "number" },
              "trendPercentage": { "type": "number" },
              "trendDirection": { "type": "string" }
            },
            "required": [
              "current",
              "previous",
              "trendPercentage",
              "trendDirection"
            ]
          },
          "quality": {
            "type": "object",
            "properties": {
              "current": { "type": "number" },
              "previous": { "type": "number" },
              "trendPercentage": { "type": "number" },
              "trendDirection": { "type": "string" }
            },
            "required": [
```

```
            "current",
            "previous",
            "trendPercentage",
            "trendDirection"
          ]
        },
        "performance": {
          "type": "object",
          "properties": {
            "current": { "type": "number" },
            "previous": { "type": "number" },
            "trendPercentage": { "type": "number" },
            "trendDirection": { "type": "string" }
          },
          "required": [
            "current",
            "previous",
            "trendPercentage",
            "trendDirection"
          ]
        }
      },
      "required": [
        "instrumentId",
        "score",
        "utilization",
        "quality",
        "performance"
      ]
    }
  },
  "page": { "type": "integer" },
  "pageSize": { "type": "integer" }
 },
 "required": ["data", "page", "pageSize"]
}
```


### 5.1.3.6 Response Example


```
{
 "data": [
  {
   "instrumentId": "Fluent~2207014657.780",
   "score": {
    "current": 0,
    "previous": 0,
    "trendPercentage": 0,
    "trendDirection": "Unchanged"
   },
   "utilization": {
```

```
      "current": 0,
      "previous": 0,
      "trendPercentage": 0,
      "trendDirection": "Unchanged"
    },
    "quality": {
      "current": 0,
      "previous": 0,
      "trendPercentage": 0,
      "trendDirection": "Unchanged"
    },
    "performance": {
      "current": 0,
      "previous": 0,
      "trendPercentage": 0,
      "trendDirection": "Unchanged"
    }
  }
 ],
 "page": 1,
 "pageSize": 27
}
```

## 5.2 Error Handling

All endpoints return appropriate HTTP status codes:
- 200 OK: Request successful
- 400 Bad Request: Invalid parameters or validation errors
- 401 Unauthorized: Authentication token missing or invalid
- 403 Forbidden: Insufficient permissions

### 5.2.1 Error Response Example


```
{
  "Errors": [
    "The timeframe between startDateTime and endDateTime cannot exceed 31 days.",
    "LabIds contains invalid or empty GUID values."
  ]
}
```

# 6 Usage Examples – Cloud API for Usage Data

## 6.1 Complete Workflow

1. Get an access token: Use the Introspect web page or refresh via the /refresh API.
   GET /api/Tokens/refresh?durationInHours=24

2. Retrieve available instruments:
   GET /api/Instruments/metadata

3. Get usage data for specific instruments:
   GET /api/Instruments/usageData?LabIds=a50e8400-e29b-41d4-0716-446655440000&InstrumentIds=InstrumentId.1&InstrumentIds=InstrumentId.2&FromDate=2025-07-08&Page=2&PageSize=10

4. Get indicator data:
   GET /api/Instruments/indicators

## 6.2 Using curl on Windows Command Line

*If curl is not installed, use choco install curl or download from https://curl.se/windows/.*

Examples:
- Refresh access token:

```
curl -L -H "Authorization: Bearer THE_TOKEN" https://cloud-api.tecan.com/api/tokens/refresh
```

- Retrieve available instruments:

```
curl -L -H "Authorization: Bearer THE_TOKEN" https://cloud-api.tecan.com/api/instruments/metadata
```

- Get usage data:

```
curl -L -H "Authorization: Bearer THE_TOKEN" https://cloud-api.tecan.com/api/instruments/usageData
```

- Get indicator data:

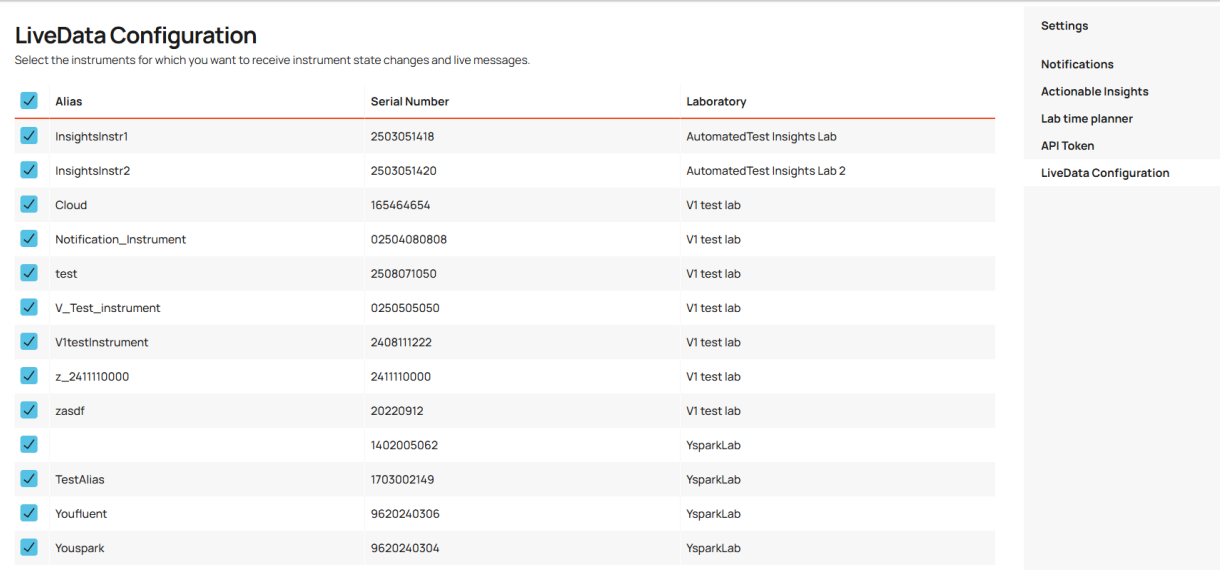
```
curl -L -H "Authorization: Bearer THE_TOKEN" https://cloud-api.tecan.com/api/instruments/indicators
```

# 7 Operation - Live Data API

## 7.1 Choice of instruments

Introspect lets you choose the instruments for which you want to get Live Data through the Live API. This can be done through the Live Data Configuration

1. Go to the Introspect Live Data Configuration page
   https://digital.tecan.com/introspect/settings/live-data




2. Enable the instruments for which you want to receive Live data.

### 7.2 Connection to the SignalR hub and handling of messages

In order to get connected to the SignalR hub, a new SignalR connection must be built and connected. In order to be able to connect, a valid token will need to be manually generated as described in 4.1
Following examples are based on a c# implementation

#### 7.2.1 Connection to the Hub

A HubConnectionBuilder is created and configured with the Token generated in 4.1:

```
var hubUrl = "https://cloud-api.tecan.com/hub/instruments";
    var jwtToken = "{your token gotten from introspect API token generation}";
    var connection = new HubConnectionBuilder()
            .WithUrl(hubUrl, options =>
            {
                    options.Headers.Add("Authorization", $"Bearer {jwtToken}");
            })
            .Build();
```

Once the ConnectionHub is created and configured, it needs to be connected:

```
connection.StartAsync()
```

#### 7.2.2 Automatic Reconnection

If you want that the HubConnection tries to reconnect automatically when connection is lost, add the following when creating it:

```
.WithAutomaticReconnect()
```

### 7.2.3 Messages

2 types of messages can be received in the hub: liveMessageReceived and instrumentStateChanged

### 7.2.4 liveMessageReceived:

When the instrument sends a Live Message to Introspect, Introspect forwards a message of type liveMessageReceived to the Hub.

#### 7.2.4.1 Schema

This message is a json message matching this schema:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "LiveMessage",
  "type": "object",
  "properties": {
    "Instrument": {
      "type": "object",
      "properties": {
        "Id": {
          "type": "string",
          "format": "uuid"
        },
        "InstrumentId": {
          "type": "string"
        }
      },
      "required": ["Id", "InstrumentId"]
    },
    "Content": {
      "type": "string"
    },
    "DateTime": {
      "type": "string",
      "format": "date-time"
    },
    "MessageType": {
      "type": "integer",
      "minimum": 0,
      "maximum": 25
    },
    "MessageSeverity": {
      "type": "integer",
      "minimum": 0,
      "maximum": 4
    },
    "StartTime": {
      "type": "string",
      "format": "date-time"
```

```
    },
    "EstimatedEndTime": {
      "type": "string",
      "format": "date-time"
    },
    "ProgressBarPercentage": {
      "type": "number"
    }
  },
  "required": [
    "Instrument",
    "Content",
    "DateTime",
    "MessageType",
    "MessageSeverity",
    "StartTime",
    "EstimatedEndTime",
    "ProgressBarPercentage"
  ]
}
```


#### 7.2.4.1.1 MessageType

MessageType field can have the following values:

0 = None
1 = InstrumentIsOffline
2 = InstrumentIsOnline
3 = InitializationCompletedSuccessfully
4 = InitializationCompletedWithErrors
5 = RunStarted
6 = RunIsPaused
7 = RunIsResumed
8 = RunCompletedSuccessfully
9 = RunCompletedWithWarnings
10 = RunCompletedWithErrors
11 = RunWasStopped
12 = UserInteractionRequired
13 = RecoveryStarted
14 = RecoveryCompleted
15 = System
16 = InstrumentName
17 = Webcam
18 = InstrumentState
19 = SoftwareVersion
20 = ControlApplicationName
21 = LoggedInUser
22 = Custom
23 = RunInfo
24 = RunName
25 = RunState

#### 7.2.4.1.2 MessageSeverity

MessageSeverity field can have the following values:

0 = Other
1 = Comment
2 = Info
3 = Warning
4 = Error

### 7.2.4.2 Example of a liveMessageReceived


```
{
  "Instrument": {
    "Id": "b3b6c1e2-8c2a-4e7a-9b1a-123456789abc",
    "InstrumentId": "Instrument-001"
  },
  "Content": "Instrument is running",
  "DateTime": "2025-12-17T12:34:56.789Z",
  "MessageType": 5,
  "MessageSeverity": 2,
  "StartTime": "2025-12-17T12:00:00.000Z",
  "EstimatedEndTime": "2025-12-17T13:00:00.000Z",
  "ProgressBarPercentage": 50.0
}
```

### 7.2.5 instrumentStateChanged

When the instrument sends a State Change to Introspect, Introspect forwards a message of type instrumentStateChanged to the Hub.

#### 7.2.5.1 Schema

This message is a json message matching this schema


```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "InstrumentState",
  "type": "object",
  "properties": {
    "Id": {
      "type": "string",
      "format": "uuid"
    },
    "State": {
      "type": "integer",
      "minimum": 0,
      "maximum": 6
    },
    "LastUpdateDateTime": {
      "type": "string",
      "format": "date-time"
    },
    "ProgressPercentage": {
      "type": "number"
    },
    "LastRunName": {
      "type": "string"
    },
    "LastRunCompletedTime": {
      "type": "string",
      "format": "date-time"
    },
    "LastErrorTime": {
      "type": "string",
      "format": "date-time"
    },
    "EstimatedRunCompleteTime": {
      "type": "string",
      "format": "date-time"
    },
    "LastStateChangeTime": {
      "type": "string",
      "format": "date-time"
    },
    "LastConnectedTime": {
      "type": "string",
      "format": "date-time"
```

```
    }
  },
  "required": [
    "Id",
    "State",
    "LastUpdateDateTime",
    "ProgressPercentage",
    "LastRunName",
    "LastRunCompletedTime",
    "LastErrorTime",
    "EstimatedRunCompleteTime",
    "LastStateChangeTime",
    "LastConnectedTime"
  ]
}
```


#### 7.2.5.1.1 InstrumentState

InstrumentState field can have the following values:

0=NotConnected
1=Available
2=Running
3=ActionRequired
4=Issues
5=Paused
6=Stopped

### 7.2.5.2 Example of a instrumentStateChanged Message


```
{
  "Id": "b3b6c1e2-8c2a-4e7a-9b1a-123456789abc",
  "State": "Running",
  "LastUpdateDateTime": "2025-12-17T12:34:56.789Z",
  "ProgressPercentage": 75.5,
  "LastRunName": "Run 42",
  "LastRunCompletedTime": "2025-12-17T11:00:00.000Z",
  "LastErrorTime": "2025-12-17T10:00:00.000Z",
  "EstimatedRunCompleteTime": "2025-12-17T13:00:00.000Z",
  "LastStateChangeTime": "2025-12-17T12:00:00.000Z",
  "LastConnectedTime": "2025-12-17T12:30:00.000Z"
}
```


## 7.3 Safety

No specific safety instructions are required for API usage. Ensure secure handling of authentication tokens.

# 8 Usage Examples – Live Data API

## 8.1 Complete Workflow

1. Get an access token: Use the Introspect web page
2. Create and configure a Hub Connection
3. Connect the HubConnection
4. Connect the different messages and events to your implementation

## 8.2 Short example of a console application

The following implementation example shows a very simple c# implementation of a console application that create a HubConnection, opens the connection and hooks some events (including the messages that can be received through the hub)
The application will need

- Microsoft.AspNetCore.SignalR.Client
- System.IdentityModel.Tokens.Jwt

```
1   using System.Text.Json;
2   using Microsoft.AspNetCore.SignalR.Client;
3
4   namespace SignalR;
5
6   public class Program
7   {
8           private static async Task Main(string[] args)
9           {
10                  Console.WriteLine("SignalR Connection Test");
11                  Console.WriteLine("======================");
12                  Console.WriteLine();
13
14                  var hubUrl = "https://cloud-api.tecan.com/hub/instruments";
15                  var jwtToken = "your token gotten from introspect API token generation";
16                  var connection = new HubConnectionBuilder()
17                          .WithUrl(hubUrl, options =>
18                          {
19                                  options.Headers.Add("Authorization", $"Bearer {jwtToken}");
20                          })
21                          .WithAutomaticReconnect()
22                          .Build();
23
24                  connection.Closed += async error =>
25                  {
26                          Console.WriteLine($"Connection closed: {error?.Message}");
27                          await Task.Delay(5000);
28                  };
29
30                  connection.Reconnecting += error =>
31                  {
32                          Console.WriteLine($"Connection reconnecting: {error?.Message}");
33                          return Task.CompletedTask;
34                  };
35
36                  connection.Reconnected += connectionId =>
```

```
37                      {
38                              Console.WriteLine($"Connection reconnected: {connectionId}");
39                              return Task.CompletedTask;
40                      };
41
42                      try
43                      {
44                              Console.WriteLine("Connecting to SignalR hub...");
45                              await connection.StartAsync();
46                              Console.WriteLine($"Connected successfully! Connection ID:
    {connection.ConnectionId}");
47
48                              connection.On<object>("liveMessageReceived", message =>
49                              {
50                                      Console.WriteLine($"Live Message Received:
    {JsonSerializer.Serialize(message)}");
51                              });
52
53
54              connection.On<object>("instrumentStateChanged", message =>
55                              {
56                                      Console.WriteLine($"Instrument State Changed Received:
    {JsonSerializer.Serialize(message)}");
57                              });
58
59
60                              Console.WriteLine("\nConnection established. Press 'q' to quit or 's' to send
    test message:");
61
62                              while (true)
63                              {
64                                      var key = Console.ReadKey(true);
65
66                                      if (key.KeyChar is 'q' or 'Q')
67                                      {
68                                              break;
69                                      }
70                              }
71                      }
72                      catch (Exception ex)
73                      {
74                              Console.WriteLine($"Connection failed: {ex.Message}");
75                      }
76                      finally
77                      {
78                              if (connection.State == HubConnectionState.Connected)
79                              {
80                                      Console.WriteLine("Disconnecting...");
81                                      await connection.DisposeAsync();
82                                      Console.WriteLine("Disconnected");
83                              }
84                      }
85
86                      Console.WriteLine("Press any key to exit...");
87                      Console.ReadKey();
88              }
89      }
```

# 9 Support

For support, contact Tecan technical support or refer to the official documentation.

# 10 Appendix

- All date/time values must be in ISO 8601 format.
- Pagination is supported for data retrieval endpoints.
- Default parameter values are applied if not specified.
- Token duration is configurable (default: 1 hour).
- All endpoints require proper authentication and authorization.