GE VERNOVA

DIGITAL

PROFICY PLANT APPLICATIONS

REST API

Proprietary Notice
The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of General Electric Company. Information contained herein is subject to change without notice.

© 2023, General Electric Company. All rights reserved.

Trademark Notices
GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:
doc@ge.com

Contents

Chapter 1. Plant Applications REST API

About Plant Applications REST API

Plant Applications is a unique software solution that digitizes the collective information being generated throughout your production facilities into a “virtual plant” for access where, when, and how you need it. Plant Applications provides clear insight into your production to greatly improve operational effectiveness.

Plant Applications REST interface provides capabilities to access Plant Applications remotely through web interface (https) which allows to build client applications on any operating system or browser-based client applications using the hypermedia provided by Plant Applications.

You can use Plant Applications REST API to create, manipulate, and search data in Plant Applications by sending HTTP requests to endpoints in Plant Applications.

Depending on where you send requests, you access and operate on different pieces of information, called resources. Resources include records, query results, metadata, and more.

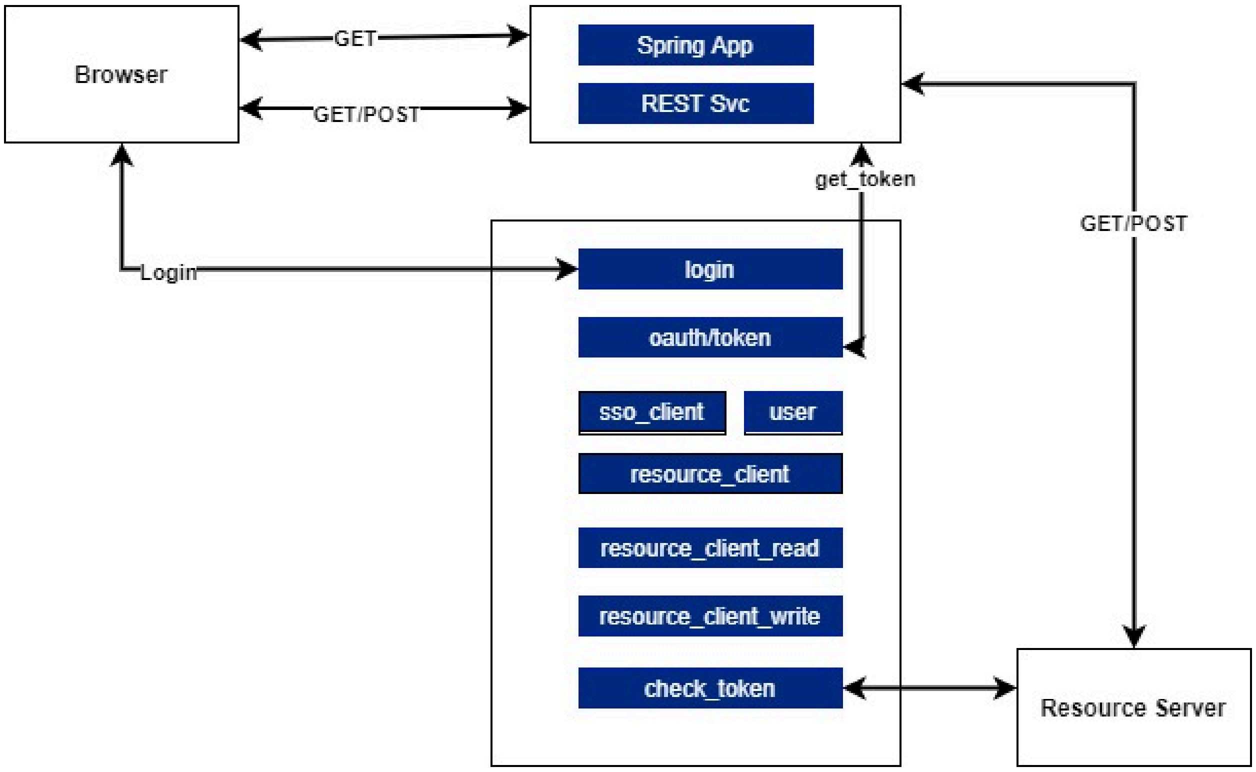
Plant Applications REST API uses RESTful architecture to provide a straightforward and consistent interface.

This document provides links for setting up your development environment, as well as information for getting started with the Plant Applications services and their associated APIs.

About Security and Authentication

For security purposes, Plant Applications uses the Proficy Authentication service as a trusted source of tokens issued for authentication. It is a multi-tenant identity management service, used in Cloud Foundry, but also available as a standalone OAuth2 server. Its primary role is as an OAuth2 provider, issuing tokens for client applications to use when they act on behalf of Cloud Foundry users. It can also authenticate users with Cloud Foundry credentials, and can act as an SSO service using those credentials, or others. It contains endpoints for managing user accounts, registering OAuth2 clients, and other management functions.

The following diagram shows how the Proficy Authentication server functions:



Access the REST APIs

About this task

The Plant Applications Web Client provides a Swagger-based UI to view and run the Representational State Transfer (REST) APIs.

Starting Plant Applications 2022, the default https port is 443. If you use the default port, you need not include it in the Rest API calls.

You can access the UI from the list of supported Web browsers by entering a URL in the following format: `https://<server_name>:<port_number>/<micro_service_name>/swagger-ui.html`.

Where:

- `<server_name>`: Represents the name of the server on which the Plant Applications Web Client is installed.
- `port_number`: Represents the network port used by the Plant Applications Web Client.

Note:

By default, the Web Client installs on port 443. When port 443 is not available, then the Web Client tries to install on port 5059. If you are installing Operations Hub and Plant Applications Web Client on the same machine, then Web Client will be installed on port 5059. If the Web Client is running on 443, then you do not need to specifically provide the port number while accessing the URL. For example, `https://<server_name>/<micro_service_name>/swagger-ui.html`.

If the Web client is running on 5059, then you must provide the port number in the URL. For example, `https://<server_name>:5059/micro_service_name>/swagger-ui.html`.

- `<micro_service_name>`: Represents the name of the microservice for which you want to access the REST APIs. See Swagger URLs *(on page 6)*.

For exchanging data between the client-server system, user authentication is required.

Procedure

1. Access the following URL: https://<server name of web client>:<port number>/<application service name>/swagger-ui.html
    - **For Workorder Service**: https://webclientservername:5059/workorder- service/apidocs/index.html
    - **For Esignature-app-service Service**: https:// webclientservername:<port>/esignature-app-service/swagger-ui/

   The Swagger UI appears.
2. To access the Swagger UIs, you must perform following steps in the Operations Hub Server:
    a. Navigate to the `<Installation Drive>:\ProgramData\GE\Operations Hub\uaa-config location`.
    b. Using a text editor, update the `uaa.yml` file by adding the below lines at the end of file with proper indentation.

```
cors:
  xhr:
    allowed:
      headers:
        - X-Requested-With
        - Authorization
      methods:
       - POST
```

c. Restart the GE Operations Hub UAA Tomcat Web Server service.

3. To access the user interface for any required service for example, Activities App service, enter the following URL in a compatible web browser: https://<server_name>:<port_number>/<micro_service_name>/swagger-ui.html.
   The Swagger UI appears.
4. Select **Authorize**.
   The **Available authorizations** page appears.
5. Enter the following values, and select **Authorize**.

Field	Description
User Name	Enter the Plant Applications Web Client login user name.
Password	Enter the Plant Applications Web Client login password.
client_id	Enter a client id value that was used during the installation.
client_secret	Enter the password. This password is set during the Web Client installation.

Swagger URLs of Rest Services

By default the Web Client installs on port 443. When port 443 is not available, then the Web Client tries to install on port 5059.

Note:
For Work Order service, you must configure security configuration (roles and assignment) in the Plant Applications Web Client before you do post API URL. For more information, see 'Create a Role' and 'Create an Assignement' in Security.

Access the swagger URLs of rest services by typing the URL in the following format: `https://<FQDN or Hostname>:<port_number>/micro_service_name/swagger-ui.html`. For example,

```
https://1[redacted]:5059/productionmetrics-app-service/swagger-ui.html
```

Sr.No	Service Name	Swagger URLs
1	access-control-service	`https://<FQDN or Hostname>:5059/access-control-ser-vice/swagger-ui.html`
2	activities-app-service	`https://<FQDN or Hostname>:5059/activities-app-ser-vice/swagger-ui.html`
3	activities-service	`https://<FQDN or Hostname>:5059/activities-service/swag-ger-ui.html`
4	alarm-app-service	`https://<FQDN or Hostname>:5059/alarm-app-service/swag-ger-ui.html`
5	alarm-service	`https://<FQDN or Hostname>:5059/alarm-service/swag-ger-ui.html`
6	approval-cockpit-app-service	`https://<FQDN or Hostname>:5059/approval-cockpit-app-service/swagger-ui.html`
7	approval-cockpit-service	`https://<FQDN or Hostname>:5059/approval-cockpit-ser-vice/swagger-ui.html`
8	assignment-service	`https://<FQDN or Hostname>:5059/assignment-service/swag-ger-ui.html`
9	bom-management-app-service	`https://<FQDN or Hostname>:5059/bom-management-app-ser-vice/swagger-ui.html`
10	comment-app-service	`https://<FQDN or Hostname>:5059/comment-app-ser-vice/swagger-ui.html`
11	comment-service	`https://<FQDN or Hostname>:5059/comment-service/swag-ger-ui.html`
12	document-management-ser-vice	`https://<FQDN or Hostname>:5059/document-management-ser-vice/swagger-ui.html`
13	downtime-app-service	`https://<FQDN or Hostname>:5059/downtime-app-ser-vice/swagger-ui.html`

Sr.No	Service Name	Swagger URLs
14	downtime-service	`https://<FQDN or Hostname>:5059/downtime-service/swag-ger-ui.html`
15	erp-export-service	`https://<FQDN or Hostname>:5059/erp-export-service/swag-ger-ui.html`
16	erp-import-service	`https://<FQDN or Hostname>:5059/erp-import-service/swag-ger-ui.html`
17	erp-scheduler-service	`https://<FQDN or Hostname>:5059/erp-scheduler-ser-vice/swagger-ui.html`
18	erp-transformation-service	`https://<FQDN or Hostname>:5059/erp-transformation-ser-vice/swagger-ui.html`
19	esignature-app-service	`https://<FQDN or Hostname>:5059/esignature-app-ser-vice/swagger-ui.html`
20	esignature-service	`https://<FQDN or Hostname>:5059/esignature-service/swag-ger-ui.html`
21	external-config-app-service	`https://<FQDN or Hostname>:5059/external-config-app-ser-vice/swagger-ui.html`
22	external-config-service	`https://<FQDN or Hostname>:5059/external-config-ser-vice/swagger-ui.html`
23	labor-service	`https://<FQDN or Hostname>:5059/labor-service/swag-ger-ui.html`
24	mes-dataservice	`https://<FQDN or Hostname>:5059/mes-dataservice/swag-ger-ui.html`
25	mes-service	`https://<FQDN or Hostname>:5059/mes-service/swag-ger-ui.html`
26	mymachines-service	`https://<FQDN or Hostname>:5059/mymachines-service/swag-ger-ui.html`
27	nonconformance-app-service	`https://<FQDN or Hostname>:5059/nonconformance-app-ser-vice/swagger-ui.html`
28	nonconformance-service	`https://<FQDN or Hostname>:5059/nonconformance-ser-vice/swagger-ui.html`

Sr.No	Service Name	Swagger URLs
29	operator-app-service	`https://<FQDN or Hostname>:5059/operator-app-service/swagger-ui.html`
30	pa-mymachines-service	`https://<FQDN or Hostname>:5059/pa-mymachines-service/swagger-ui.html`
31	plant-execution-service	`https://<FQDN or Hostname>:5059/plant-execution-service/swagger-ui.html`
32	process-order-service	`https://<FQDN or Hostname>:5059/process-order-service/swagger-ui.html`
33	processanalyzer-service	`https://<FQDN or Hostname>:5059/processanalyzer-service/swagger-ui.html`
34	product-service	`https://<FQDN or Hostname>/product-service/swagger-ui.html`
35	productionmetrics-app-service	`https://<FQDN or Hostname>:5059/productionmetrics-app-service/swagger-ui.html`
36	productionmetrics-service	`https://<FQDN or Hostname>:5059/productionmetrics-service/swagger-ui.html`
37	productionscheduler-app-service	`https://<FQDN or Hostname>:5059/productionscheduler-app-service/swagger-ui.html`
38	property-definition-app-service	`https://<FQDN or Hostname>:5059/property-definition-app-service/swagger-ui.html`
39	property-definition-service	`https://<FQDN or Hostname>:5059/property-definition-service/swagger-ui.html`
40	reasons-service	`https://<FQDN or Hostname>:5059/reasons-service/swagger-ui.html`
41	receiving-inspection-app-service	`https://<FQDN or Hostname>:5059/receiving-inspection-app-service/swagger-ui.html`
42	receiving-inspection-service	`https://<FQDN or Hostname>:5059/receiving-inspection-service/swagger-ui.html`
43	route-app-service	`https://<FQDN or Hostname>:5059/route-app-service/swagger-ui.html`

Sr.No	Service Name	Swagger URLs
44	route-service	`https://<FQDN or Hostname>:5059/route-service/swagger-ui.html`
45	security-administration-app-service	`https://<FQDN or Hostname>:5059/security-administration-app-service/swagger-ui.html`
46	security-service	`https://<FQDN or Hostname>:5059/security-service/swagger-ui.html`
47	segments-definition-service	`https://<FQDN or Hostname>:5059/segments-definition-service/swagger-ui.html`
48	spc-app-service	`https://<FQDN or Hostname>:5059/spc-app-service/swagger-ui.html`
49	supervisor-app-service	`https://<FQDN or Hostname>:5059/supervisor-app-service/swagger-ui.html`
50	time-booking-app-service	`https://<FQDN or Hostname>:5059/time-booking-app-service/swagger-ui.html`
51	usersettings-service	`https://<FQDN or Hostname>:5059/usersettings-service/swagger-ui.html`
52	waste-management-app-service	`https://<FQDN or Hostname>:5059/waste-management-app-service/swagger-ui.html`
53	waste-management-service	`https://<FQDN or Hostname>:5059/waste-management-service/swagger-ui.html`
54	webgenealogy-app-service	`https://<FQDN or Hostname>:5059/webgenealogy-app-service/swagger-ui.html`
55	work-order-history-service	`https://<FQDN or Hostname>:5059/work-order-history-service/swagger-ui.html`
56	work-order-service	`https://<FQDN or Hostname>:5059/work-order-service/api-docs/index.html`
57	operator-app-log-service	`https://<FQDN or Hostname>:5059/operator-app-log-service/swagger-ui.html`
58	operator-log-service	`https://<FQDN or Hostname>:5059/operator-log-service/swagger-ui.html`

Sr.No	Service Name	Swagger URLs
59	line-overview-service	`https://<FQDN or Hostname>:5059/line-overview-service/swagger-ui.html`
60	line-overview-app-service	`https://<FQDN or Hostname>:5059/line-overview-app-service/swagger-ui.html`
61	autolog-service	`https://<FQDN or Hostname>:5059/autolog-service/swagger-ui.html`
	autolog-app-service	`https://<FQDN or Hostname>:5059/autolog-app-service/swagger-ui.html`

REST Resources and Requests

Plant Applications REST API is based on the use of resources or pieces of data in Plant Applications, such as:

- Records
- List of records
- Query results
- Metadata
- API information

Each resource is exposed by a uniform resource locator (URL) and is accessed by sending HTTP requests to the corresponding URL.

Depending on which resource you want to access and how you construct an HTTP request, you can perform the following:

- Determine available API versions
- Create, read, update, and delete records
- Query and search data

A typical HTTP request includes:

- URL
- HTTP method
- Headers
- Request body

URLs

You can use the Swagger documentation URLs to view all the endpoints. The Plant Applications Web Client provides a Swagger-based UI to view and run the REST APIs.

Access the Swagger UI for the microservices by using the following URL: https://<server name of web client>:<port number>/<application service name>/swagger-ui.html.

Where:

<server_name>: Represents the name of the server on which the Plant Applications Web Client is installed.

<port_number>: Represents the network port used by the Plant Applications Web Client.

<micro_service_name>: Represents the name of the microservice for which you want to run the REST APIs.

HTTP Methods

Plant Applications REST API supports these standard HTTP request methods:

- GET
- POST
- PUT
- DELETE

Method	Usage	Resource Examples
GET (Read)	Retrieves a resource.	/Waste locations **Purpose**: Get locations of Waste. **Return**: A collection of Waste locations. **Status Code**: • 200 Successful retrieval of all Waste locations. • 401 Unauthorized. • 404 Record Not Found.

Method	Usage	Resource Examples
POST (Create)	Creates or adds a resource.	/Waste Event Record **Purpose**: Create Waste Event Records. **Return**: Creation of Waste Event Records. **Status Code**: • 201 Created. • 401 Unauthorized (No Content). • 403 Forbidden. • 400 Invalid data supplied.
PUT (Update)	Updates a resource.	/Waste Events **Purpose**: Bulk update of Waste Events. **Return**: Waste Events updated. **Status Code**: 204 No Content (success) • 201 Created. • 400 Invalid data supplied. • 401 Unauthorized. • 404 Not Found. • 422 Unprocessable Entity.
DELETE (Delete)	Removes a resource.	/Waste Events **Purpose**: Delete Waste Events by ID. **Return**: Waste Events Deleted. **Status Code**:

Method	Usage	Resource Examples
		• 200 OK. • 400 Invalid data supplied. • 401 Unauthorized. • 403 Forbidden. • 404 Not Found.

HTTP Status Codes

Each request to the REST API results into a standard HTTP status code. When using the following HTTP status codes, the Plant Applications API services adhere as closely as possible to standard HTTP and REST conventions.

When working with REST APIs, the status code for every HTTP request to the server is returned according to the HTTP status code described in the RFC 2616 and RFC 6585. The HTTP codes are grouped into five different categories:

Table 1.

Status Code	Meaning	Comments
1xx	Informational	Provisional response.
2xx	Successful	Clients request was successfully received, understood and accepted.
3xx	Redirection	Further action needs to be taken by the user.
4xx	Client error	Client did something wrong. It should NOT send the same request again, but fix the issue first.
5xx	Server error	Server did something wrong or incapable of performing the request. Client can continue and try again with the same request.

The HTTP status codes clearly indicate the status for every REST API response:

Status Code	Usage
200 OK	Success message. The request has been completed.

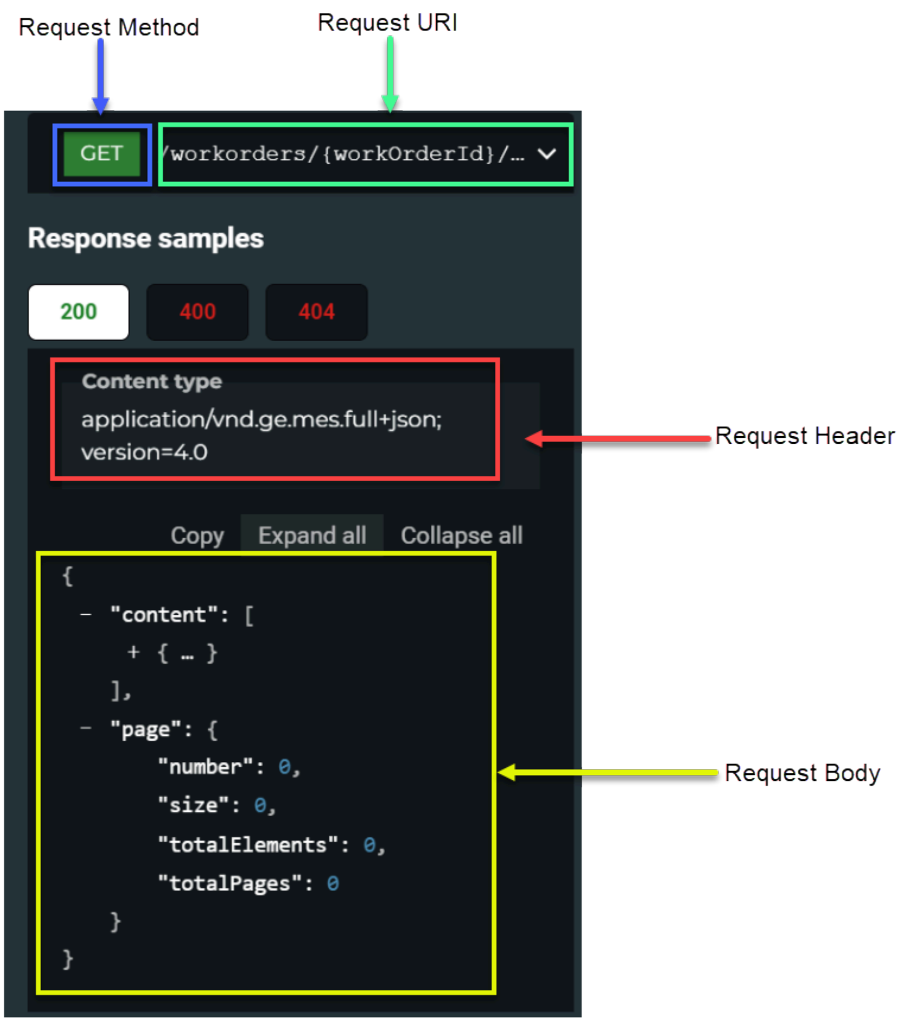
Status Code	Usage
201 Created	Success message. A new resource has been created. The resource URI is available from the location header in the response.
204 No Content	Success message. An update to an existing resource has been applied.
400 Bad Request	Error message. The request was malformed. The response body provides additional information.
401 Unauthorized	Error message. Either you are not authenticated, or the authentication is incorrect. You must re-authenticate and try again.
403 Forbidden	Error message. You do not have permission to access this resource.
404 Not Found	Error message. The requested resource does not exist.
409 Conflict	Error message. The request conflicts with another request.
500 Internal Error	Error message. The server encountered an unexpected condition that prevented it from fulfilling the request.

Headers

You can use headers to pass parameters and customize options for HTTP requests. Plant Applications REST API supports several standard HTTP headers. The common headers are:

- HTTP Accept - Indicates the format that your client accepts for the response body. The default value is application/json.
- HTTP Content-type - Indicates the format of the request body that you attach to the request.
- HTTP Authorization - Provides the OAuth 2.0 access token to authorize your client.

Below is an example of a sample header:



Content type
application/vnd.ge.mes.full+json; version=4.0

```
{
  - "content": [
      + { … }
    ],
  - "page": {
        "number": 0,
        "size": 0,
        "totalElements": 0,
        "totalPages": 0
    }
}
```

Request Bodies

A request body is a rich input that can be included in the request to provide additional information, such as field values for creating or updating records. A request body is in JSON format.

Errors

Plant Applications microservices frequently return an error response for a REST request but often the standard HTTP status codes are insufficient to provide enough information for a user to correct the error.

This section describes an example of a standard error response body structure for work-order-service that can be used to provide additional arbitrary details back to the user.

Detailed Error Response Format

The error response format consists of application/JSON data with the following fields:

`Timestamp` (string) Timestamp of error in ISO8601 format.

Path (string) HTTP request path where the error occurred.

`Error` (class) A single error structure consisting of the following fields:

- `Code` (string) Enumeration value of the error classification or type.
- `Details` (class) An optional class (could be null) containing arbitrary details about the specific error occurrence.

A user issuing a request that results in an error receives a response of the above format and expects a known Details structure for any given error Code. If the error is correctable, then the data stored within the Details provides sufficient knowledge required to correct the error and reissue the request. If the error is not correctable then the Details will be empty.

Examples

This section lists out few examples on standard errors such as:

- 404 Not Found *(on page 17)*
- 422 Unprocessable Entity *(on page 18)*
- 409 Conflict *(on page 18)*
- 403 Forbidden *(on page 19)*
- 400 Bad Request *(on page 19)*

404 Not Found

```
404 Not Found

{

    timestamp: "2017-03-21T15:25:21+00:00"

    path: "/example/1"

    error: {

        code: "ResourceNotFound"
    }
}
```

422 Unprocessable Entity

```
422 Unprocessable Entity
{
    timestamp: "2017-03-21T15:25:21+00:00"
    path: "/productionratios"
    error: {
        code: "ContentDataError"
        details: {
            [
                {
                    lineNumber: 3,
                    errorCode: DuplicateUnit
                },
                {
                    lineNumber: 7,
                    errorCode: DuplicateUnit
                },
                {
                    lineNumber: 8,
                    errorCode: ValueOutOfRange
                }
            ]
        }
    }
}
```

409 Conflict

```
409 Conflict
{
       "timestamp": "2017-03-17T17:48:23Z",
       "path": "https://localhost/mes-service/v1/downtimeRecords"
       "error": {
              "code": "OpenDowntimeRecordConflict",
              "details": {
                      "requestedStartTime": "2017-03-17T17:48:23Z",
                      "requestedEndTime": "2017-03-17T17:48:23Z",
                      "openDowntimeRecordId": 342562,
                      "openStartTime": "2017-03-17T17:48:23Z",
                      "openEndTime": "2017-03-17T17:48:23Z"
              }
       }
}
```

403 Forbidden

```
403 Forbidden
{
       "timestamp": "2017-03-17T17:48:23Z",
       "path": "https://localhost/mes-service/v1/downtimeRecords/25143"
       "error": {
              "code": "InsufficientPermission",
              "details": null
       }
}
```

400 Bad Request

```
400 Bad Request
{
       "timestamp": "2017-03-17T17:48:23Z",
       "path": "https://localhost/mes-service/v1/downtimeRecords/25143"
       "error": {
              "code": "MissingRequiredData",
              "details": {
                      "propertyName": "comment"
              }
       }
}
```

Standard

Plant Applications REST API follows the standard RESTful principles and characteristics:

- Client-server: Client applications are independent from Plant Applications REST API, meaning each is managed and updated independently.
- Stateless: Each request from client to server must contain all the information necessary to understand the request, and not use any stored context on the server. However, the representations of the resources are interconnected using URLs, which allow the client to progress between states.
- Caching behavior: Responses are labeled as cacheable or non-cacheable.
- Uniform interface: All resources are accessed with a generic interface over HTTPS.
- Layered System: Intermediaries, such as proxy servers and gateways, are allowed between the client and the resources.
- Authentication: Plant Applications REST API supports OAuth 2.0 (an open protocol to allow secure API authorization).

REST API EndPoints

This section includes a list of supported Plant Applications REST API endpoints.

Common Services

- access-control-service
- activities-app-service
- activities-service
- alarm-app-service
- alarm-service
- comment-app-service
- comment-service
- downtime-app-service
- downtime-service
- erp-import-service
- erp-transformation-service
- erp-export-service
- erp-scheduler-service
- mes-dataservice
- mes-service
- mymachines-service

- pa-mymachines-service
- product-service
- productionmetrics-app-service
- productionmetrics-service
- reasons-service
- security-administration-app-service
- spc-app-service
- security-service
- user-settings-service

Process Services

- operator-log-service
- operator-log-app-service
- line-overview-service
- line-overview-app-service
- autolog-service
- autolog-app-service
- esignature-app-service
- esignature-service
- process-order-service
- productionscheduler-app-service
- waste-management-app-service
- waste-management-service

Discrete Services

- approvalcockpit-app-service
- approval-cockpit-service
- assignment-service
- bom-management-app-service
- document-management-service
- external-config-app-service
- external-config-service
- labor-service
- nonconformance-app-service
- nonconformance-service
- operator-app-service

- plant-execution-service
- property-definition-app-service
- property-definition-service
- receiving-inspection-app-service
- receiving-inspection-service
- route-app-service
- route-service
- segments-definition-service
- supervisor-app-service
- time-booking-app-service
- webgenealogy-app-service
- work-order-history-service
- work-order-service

Examples

This section provides examples for using Plant Applications REST API resources to do a variety of different tasks such as:

- Query Activities in a Certain State *(on page 22)*.
- Query Serials or Batches Produced on a Unit for a Given Time *(on page 23)*.
- Query Units and their Variable Details *(on page 25)*.
- Query Production Events *(on page 26)*.
- View the Specification for a Batch, Serial, or Product of an Equipment *(on page 28)*.

Query Activities in a Certain State

About this task

The following example shows you how to:

- Retrieve information on activities in a certain state.

Procedure

1. Log in to the Swagger UI for activities-app service.
   The **Activities App service** swagger page appears.
2. Select **activity-viewer-controller**.

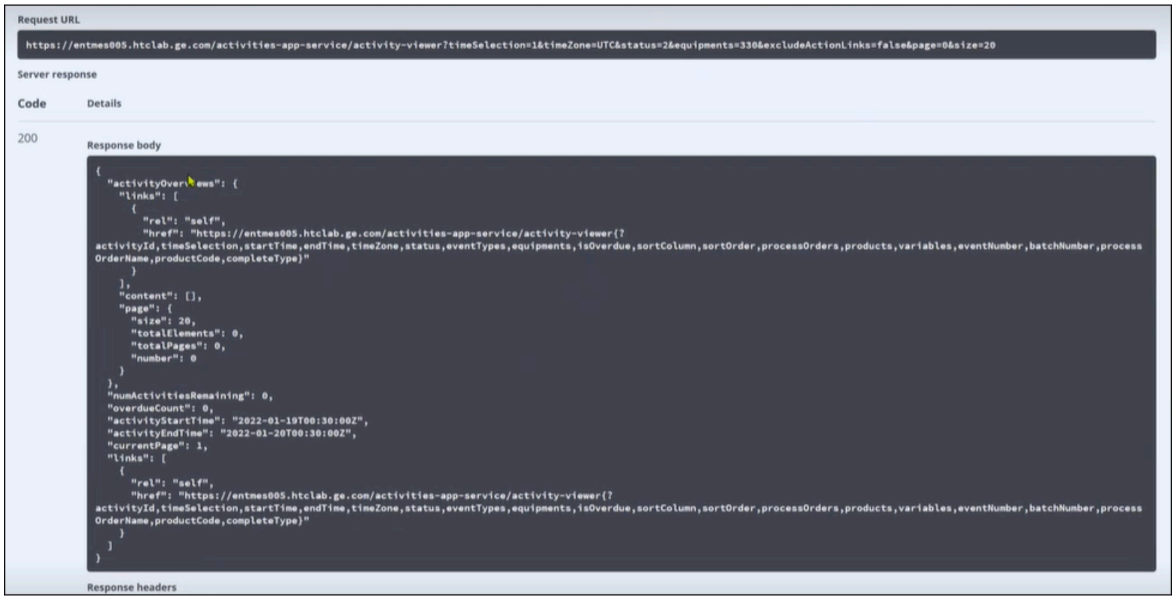
   All the methods for the production-event-controller appear.

3. Select **GET/activity-viewer** to fetch all the records related to the activity.
4. Select **Try it out**, then enter the following query parameters:
    a. Activity id or a list of activity ids for which you want to search the data for.
    b. Time selection values to display activities for the duration. For example, Current Day, Current Week, Next Day, Next Week, and so on.
    c. Start time and end time of the activity.
    d. Status to filter activities in a certain state. For example, Not Started, Incomplete, and so on.
5. Select **Execute**.

    The following details for the given Activity id and status appears:

    ◦ Activity ID
    ◦ The Start time and End time of the activity
    ◦ The activity status.

    The following shows an example of the response, displaying activities in a certain state.



Request URL

```
https://entmes005.htclab.ge.com/activities-app-service/activity-viewer?timeSelection=1&timeZone=UTC&status=2&equipments=330&excludeActionLinks=false&page=0&size=20
```

Server response

Code Details

200 Response body

```
{
  "activityOverviews": {
    "links": [
      {
        "rel": "self",
        "href": "https://entmes005.htclab.ge.com/activities-app-service/activity-viewer{?
activityId,timeSelection,startTime,endTime,timeZone,status,eventTypes,equipments,isOverdue,sortColumn,sortOrder,processOrders,products,variables,eventNumber,batchNumber,process
OrderName,productCode,completeType}"
      }
    ],
    "content": [],
    "page": {
      "size": 20,
      "totalElements": 0,
      "totalPages": 0,
      "number": 0
    }
  },
  "numActivitiesRemaining": 0,
  "overdueCount": 0,
  "activityStartTime": "2022-01-19T00:30:00Z",
  "activityEndTime": "2022-01-20T00:30:00Z",
  "currentPage": 1,
  "links": [
    {
      "rel": "self",
      "href": "https://entmes005.htclab.ge.com/activities-app-service/activity-viewer{?
activityId,timeSelection,startTime,endTime,timeZone,status,eventTypes,equipments,isOverdue,sortColumn,sortOrder,processOrders,products,variables,eventNumber,batchNumber,process
OrderName,productCode,completeType}"
    }
  ]
}
```

Response headers

Query Serials or Batches Produced on a Unit for a Given Time

About this task

The following example shows you how to:

- Retrieve information on serials or batches produced on a unit for a given time.

Procedure

1. Log in to the Swagger UI for activities-app service.

   The **Activities App service** swagger page appears.
2. Select **activity-viewer-controller**.

   All the methods for the activity-viewer-controller appear.
3. Select **GET/activity-viewer Fetches the activity records** to fetch all the details related to the activity.
4. Select **Try it out**, then enter the following:
   a. Time selection values to display activities for the duration. To retrieve activities for a specific time duration, use the custom time selection, and enter the startTime and endTime in ISO format. For example, ISO-8601 format (YYYY-MM-DD Thh:mm:ssZ).
   b. List of unit ids to retrieve the activities for.
   c. Status to filter activities in a certain state.
5. Select **Execute**.

   Note:

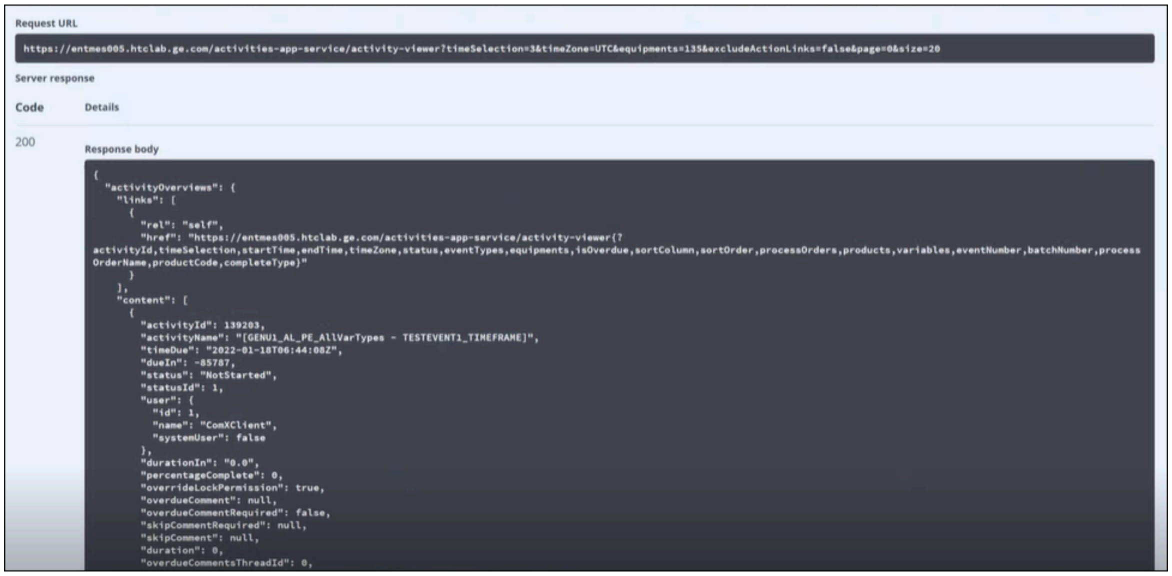
   You can filter the activity related data based on the following:

   - Event number
   - Batch number
   - Process order name
   - Product code

   The following Activity related details appear:

   - The activity id.
   - Time duration for the particular activity.
   - The activity status.
   - The Product id.

The following shows an example of the response, displaying activity related details such as serials or batches produced on unit for a given time.



Request URL

```
https://entmes005.htclab.ge.com/activities-app-service/activity-viewer?timeSelection=3&timeZone=UTC&equipments=135&excludeActionLinks=false&page=0&size=20
```

Server response

Code Details

200 Response body

```
{
  "activityOverviews": {
    "links": [
      {
        "rel": "self",
        "href": "https://entmes005.htclab.ge.com/activities-app-service/activity-viewer{?
activityId,timeSelection,startTime,endTime,timeZone,status,eventTypes,equipments,isOverdue,sortColumn,sortOrder,processOrders,products,variables,eventNumber,batchNumber,process
OrderName,productCode,completeType}"
      }
    ],
    "content": [
      {
        "activityId": 139203,
        "activityName": "[GENU1_AL_PE_AllVarTypes - TESTEVENT1_TIMEFRAME]",
        "timeDue": "2022-01-18T06:44:08Z",
        "dueIn": -85787,
        "status": "NotStarted",
        "statusId": 1,
        "user": {
          "id": 1,
          "name": "ComXClient",
          "systemUser": false
        },
        "durationIn": "0.0",
        "percentageComplete": 0,
        "overrideLockPermission": true,
        "overdueComment": null,
        "overdueCommentRequired": false,
        "skipCommentRequired": null,
        "skipComment": null,
        "duration": 0,
        "overdueCommentsThreadId": 0,
```

Query Units and their Variable Details

About this task

The following example shows you how to find variables configured on a sheet for a unit.

Procedure

1. Log in to the Swagger UI for activities-app service.

   The **Activities App service** swagger page appears.
2. Select **variable-editor-controller**.

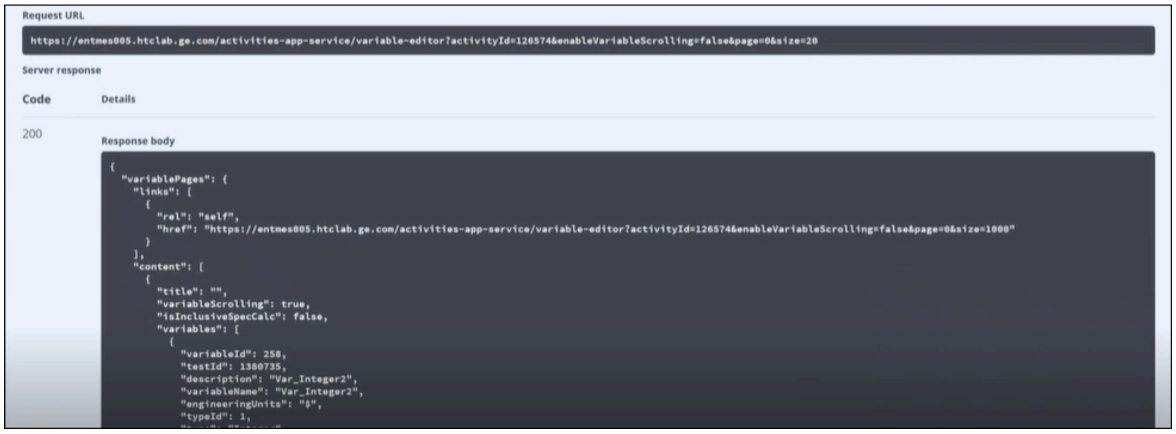
   All the methods for the variable-editor-controller appear.
3. Select **GET/variable-editor Fetches activity variables details** to fetch all the activity variables details.
4. Select **Try it out**, then enter the activity id as the query parameter.
5. Select **Execute**.

   The following variables for the required activity appear:

   - Variable id
   - Description
   - Variable name
   - Engineering unit

- ◦ Variable Type
- ◦ Source
- ◦ Specification details

The following shows an example of the response, displaying units and their variable details.



Request URL

```
https://entmes005.htclab.ge.com/activities-app-service/variable-editor?activityId=126574&enableVariableScrolling=false&page=0&size=20
```

Server response

Code Details

200 Response body

```
{
  "variablePages": {
    "links": [
      {
        "rel": "self",
        "href": "https://entmes005.htclab.ge.com/activities-app-service/variable-editor?activityId=126574&enableVariableScrolling=false&page=0&size=1000"
      }
    ],
    "content": [
      {
        "title": "",
        "variableScrolling": true,
        "isInclusiveSpecCalc": false,
        "variables": [
          {
            "variableId": 258,
            "testId": 1380735,
            "description": "Var_Integer2",
            "variableName": "Var_Integer2",
            "engineeringUnits": "$",
            "typeId": 1,
```

Query Production Events

About this task

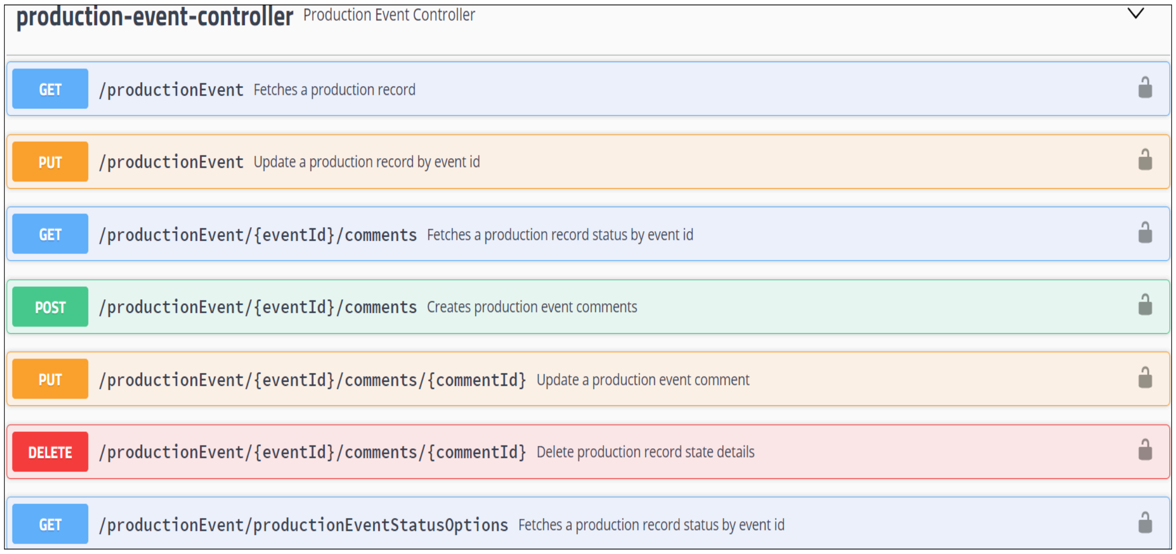
The following use case shows you how to retrieve information on production events and data related to the event.

Procedure

1. Log in to the Swagger UI for activities-app service.

   The **Activities App service** swagger page appears.
2. Select **production-event-controller**.

   All the methods for the production-event-comtroller appear.



production-event-controller Production Event Controller

GET /productionEvent Fetches a production record

PUT /productionEvent Update a production record by event id

GET /productionEvent/{eventId}/comments Fetches a production record status by event id

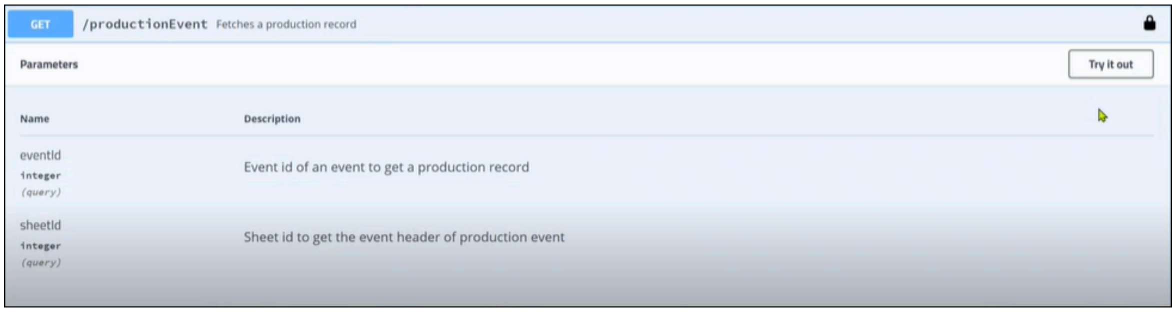
POST /productionEvent/{eventId}/comments Creates production event comments

PUT /productionEvent/{eventId}/comments/{commentId} Update a production event comment

DELETE /productionEvent/{eventId}/comments/{commentId} Delete production record state details

GET /productionEvent/productionEventStatusOptions Fetches a production record status by event id

3. Select **GET/productionEvent** call to fetch all the details related to the production event.



GET /productionEvent Fetches a production record

Parameters

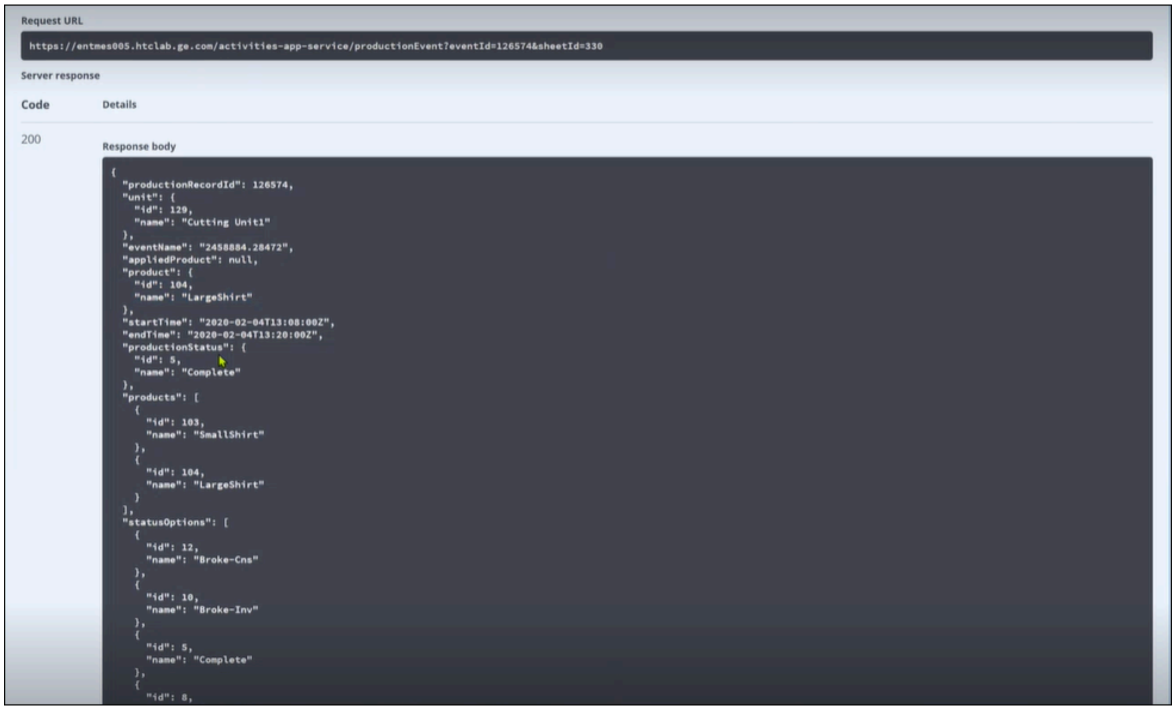
Name	Description
eventId integer (query)	Event id of an event to get a production record
sheetId integer (query)	Sheet id to get the event header of production event

4. Select **Try it out**, then enter the following parameters to query data:
   a. The Event id of an event to search for the production event data.
   b. The Sheet id to get the event header of production event.
5. Select **Execute**.

   The following details for the given event id and sheet id appears:

   - Event id on which the unit is running.
   - The name of the event.
   - The product which it was used for.
   - The start time and status of the product being manufactured.
   - The products that are manufactured for a given unit.
   - The status options for the production event for a given unit such as Consumed, Hold, Complete, and so on.
   - Comments such as dimensions, unit of measure, initial value, and so on.

The following shows an example of the response, displaying production event details.



Request URL

```
https://entmes005.htclab.ge.com/activities-app-service/productionEvent?eventId=126574&sheetId=330
```

Server response

Code Details

200 Response body

```
{
  "productionRecordId": 126574,
  "unit": {
    "id": 129,
    "name": "Cutting Unit1"
  },
  "eventName": "2458884.28472",
  "appliedProduct": null,
  "product": {
    "id": 104,
    "name": "LargeShirt"
  },
  "startTime": "2020-02-04T13:08:00Z",
  "endTime": "2020-02-04T13:20:00Z",
  "productionStatus": {
    "id": 5,
    "name": "Complete"
  },
  "products": [
    {
      "id": 103,
      "name": "SmallShirt"
    },
    {
      "id": 104,
      "name": "LargeShirt"
    }
  ],
  "statusOptions": [
    {
      "id": 12,
      "name": "Broke-Cns"
    },
    {
      "id": 10,
      "name": "Broke-Inv"
    },
    {
      "id": 5,
      "name": "Complete"
    },
    {
      "id": 8,
```

View the Specification for a Batch, Serial, or Product of an Equipment

About this task

The following use case shows you how to view specifications that are directly attached to the product that is running.

Procedure

1. Log in to the Swagger UI for activities-app service.

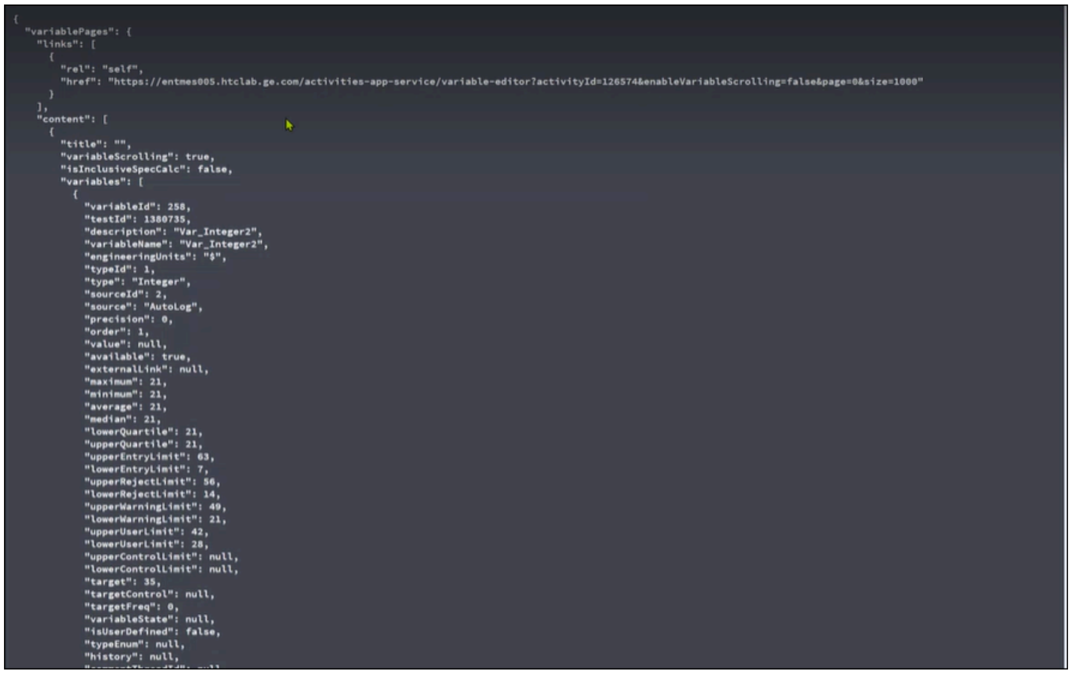
   The **Activities App service** swagger page appears.
2. Select **variable-editor-controller**.

   All the methods for the variable editor controller appear.
3. Select **GET/variable-editor Fetches activity variables details** to fetch all the details related to the variables in activity.
4. Select **Try it out**, then enter the following:
   a. Activity id for which you want to search the data for.
   b. variableFilter to filter variable data. For example, density of chamber1, chamber2, and so on.
5. Select **Execute**.

   The following details appear:

- Variable id
- Specifications such as:
    - Upper entry line
    - Lower entry line
    - Lower user limit
    - Upper user limit
    - Upper control limit
    - Lower control limit

The following shows an example of the response, displaying specification details for a batch, serial, or product of an equipment.



```
{
  "variablePages": {
    "links": [
      {
        "rel": "self",
        "href": "https://entmes005.htclab.ge.com/activities-app-service/variable-editor?activityId=126574&enableVariableScrolling=false&page=0&size=1000"
      }
    ],
    "content": [
      {
        "title": "",
        "variableScrolling": true,
        "isInclusiveSpecCalc": false,
        "variables": [
          {
            "variableId": 258,
            "testId": 1380735,
            "description": "Var_Integer2",
            "variableName": "Var_Integer2",
            "engineeringUnits": "$",
            "typeId": 1,
            "type": "Integer",
            "sourceId": 2,
            "source": "AutoLog",
            "precision": 0,
            "order": 1,
            "value": null,
            "available": true,
            "externalLink": null,
            "maximum": 21,
            "minimum": 21,
            "average": 21,
            "median": 21,
            "lowerQuartile": 21,
            "upperQuartile": 21,
            "upperEntryLimit": 63,
            "lowerEntryLimit": 7,
            "upperRejectLimit": 56,
            "lowerRejectLimit": 14,
            "upperWarningLimit": 49,
            "lowerWarningLimit": 21,
            "upperUserLimit": 42,
            "lowerUserLimit": 28,
            "upperControlLimit": null,
            "lowerControlLimit": null,
            "target": 35,
            "targetControl": null,
            "targetFreq": 0,
            "variableState": null,
            "isUserDefined": false,
            "typeEnum": null,
            "history": null,
```

Troubleshooting REST API

In order to understand the errors that you get from the Plant Applications REST API when making a request, verify the status code of the response along with the response body.

This section lists out few scenarios that you might encounter when working with the REST APIs:

- Authentication Issues Error *(on page 30)*
- REST API Displays the 403 Forbidden Error *(on page 30)*
- Cannot Access REST API over HTTPS *(on page 30)*

Authentication Issues Error

When you try to access the user interface for any required service, for example, Downtime app service, the system displays the '**401 Unauthorized Error**'. This indicates that the credentials you entered could not be authenticated using the token in the HTTP "Authorization" header.

Possible root cause: The following issues could cause the error related to authentication issues:

- The token is missing from the header or is invalid.
- The "Authorization" header is missing or is invalid.
- The token has expired or is deleted.

Resolution: The following methods can help resolve the error:

- Verify that you entered the URL correctly.
- Clear the browser's cache.
- Verify that you entered your credentials correctly.

REST API Displays the 403 Forbidden Error

The system displays the '**403 Forbidden**' error when you try to access an API and do not have the permission to perform the actions required by the API. For example, you want to create a Work Order and you do not have the permission to perform this task, then the system displays the error.

Resolution: Request the administrator to provide the permission using Security.

Cannot Access REST API over HTTPS

When you try to access the Plant Applications REST API using HTTPS, the system does not connect to the REST API.

Resolution: Configure HTTPS or enable API access over HTTP.