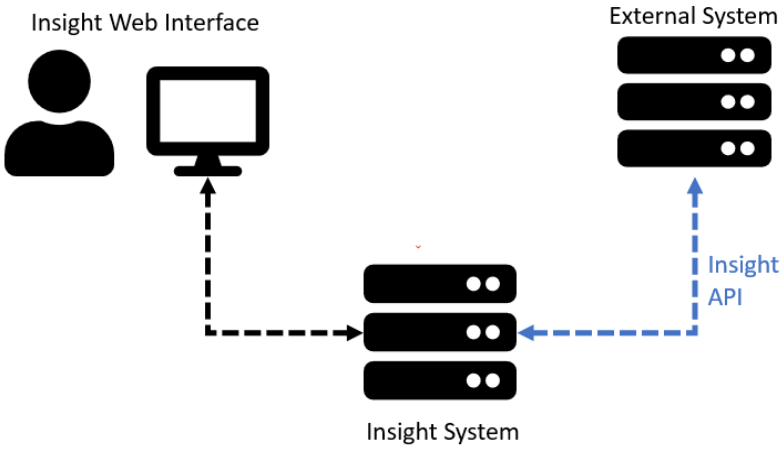
EKM Insight API

Applies to API version 1.15 and above (Portal v35)

Contents

Overview

By using the Insight API you can immediately access the powerful MPS capabilities of Insight from other systems. The Insight API provides a machine-to-machine interface that makes integration with your own systems simple.



The Insight API provides a RESTful interface using JSON over HTTPS. The API currently provides access to:

- Customers
- Monitors
- Devices
- Device Meters
- Device Consumable Levels
- Device Alerts
- Consumable Requests

Secure, role-based access to the API is fully segmented, with access keys providing views limited to individual customers, customer groups/reseller accounts, or global access.

Developers Guide

The Insight API uses REST principles to have predictable, resource-oriented URLs and uses HTTP response codes to indicate API errors.

It uses standard HTTP features, like HTTP authentication and HTTP verbs, which are understood by off-the-shelf HTTP clients. JSON is returned in all responses from the API, including errors, and is expected as the body for all requests.

Every interface is documented using OpenAPI (Swagger), and interactive documentation can be found at:

https://*your-portal-address*/PortalAPI/swagger-ui/index.html

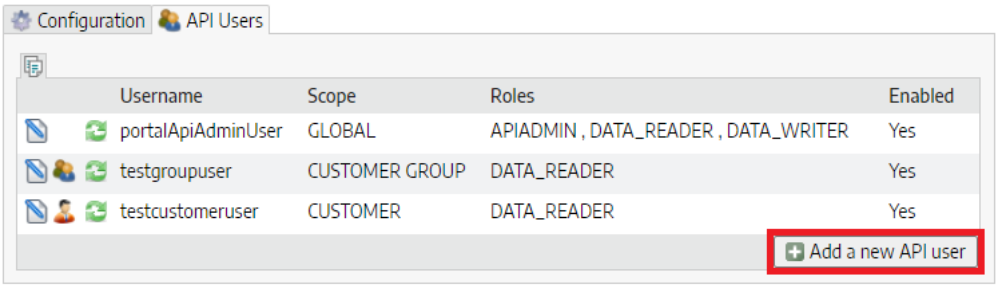
Getting Started

Accessing the Insight API is a 4-step process. This process will be familiar to any developer who has previously used REST APIs.

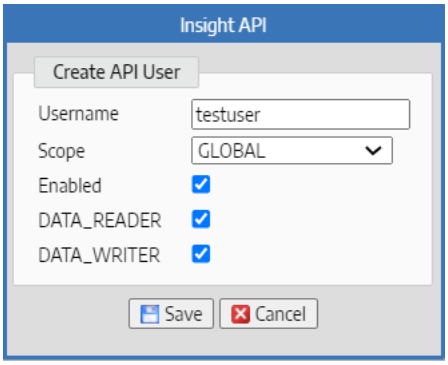
1. Obtain an API key and secret

The Insight API is managed from the Insight Portal web user interface. If you do not have access rights for Insight administration functions, ask an admin user to create an API account for you following these instructions.

Log in as a user with 'System Configuration' privileges and access the 'Insight API' section under the 'System Administration' menu. Under the 'API Users' tab, click the 'Add a new API user' button.

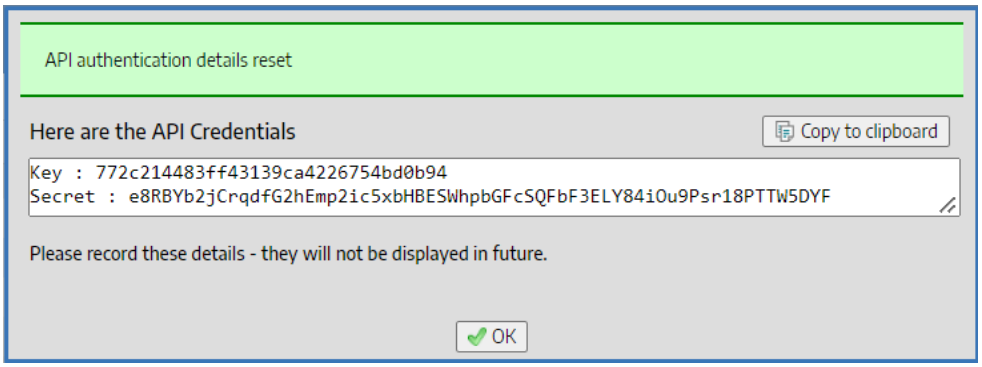


The 'Add a new API user' dialog will be displayed:



Enter a username - this is used to provide a visible name for the account and for access logging. Select the account access scope. The scope setting can be used to limit the API user's data access only to specific customers or customer groups. In this example we will use 'Global' scope, which will

give us access to all data in the Insight system. Next click ‘Save’. A new dialog will appear containing the generated API key and secret:



Take a copy of these values as they cannot be re-displayed, they can only be re-generated.

2. Encode the key

The next step is to encode the key and secret as Base64 in the form Key:Secret. This will usually be performed programmatically, however, for testing purposes there are online tools that will perform this function (https://www.base64encode.org/). The encoding process will convert the Key:Secret from this:

> *772c214483ff43139ca4226754bd0b94:e8RBYb2jCrqdfG2hEmp2ic5xbHBESWhpbGFcSQFbF3ELY84iOu9Psr18PTTW5DYF*

To this:

> *MGRmOTAxZDkyMmFlNDg1MjhmYTA0MzQ0OGZjMjNjOTY6aFQzOHY0czRPaWI5N2c5S3lpQWs5ZzJUVm96cWRYcklKYTJ2MEJvMGg1UElmSDAzbTF5aGZYY2FHWmttZlZZaw==*

3. Perform login

The next step is to use HTTP Basic authentication to perform a login, this will return a JWT access token that will provide API access for the next 24 hours. This can be accomplished by performing an HTTP POST to:

> https://*your-portal-address*/PortalAPI/login

The request should have an ‘Authorization’ header set to Basic and then the value of the Base64 encoded Key:Secret. For example:

> *Authorization: Basic MGRmOTAxZDkyMmFlNDg1MjhmYTA0MzQ0OGZjMjNjOTY…….*

The response to this request will contain the JWT access token and expiration time in seconds, for example:

> *{"access_token":"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIwNzdlMzY3MGI1OTE0NDUxOWRlMzM4YWMxNDFmNWRlYyIsImV4cCI6MTUzOTI1NDU4MX0.c-mcBlkh4w-WCBOLTQ7TmpqnK0qq-PvSwWHj5lKDY-UhT65iUBCcIG9sR1Xu0coWJZGCsJs37u1ofyPjsMhGZQ","expires_in":86400}*

4. Call API methods

We can now use the access token from the previous response to call any function on the API. The simplest request is to list customers. This can be achieved by performing an HTTP GET to:

https://*your-portal-address*/PortalAPI/api/customers/

The request should have an 'Authorization' header set to Bearer and then the value of the access_token. For example:

Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIwNzdlMzY3MGI1OTE0NDU……

The response will contain a JSON array of customers in the Insight System. For example:


```
[
  {
    "name": "ACME Corp",
    "address": "123 Long Road",
    "city": "Austin",
    "zip": "73301",
    "country": "US",
    "contactName": "W. Coyote",
    "contactEmail": "wcoyote@acmecorp.com",
    "telephoneNumber": "555-356287",
    "status": "ACTIVE",
    "customerId": 252,
    "groupId": 1
  },
  {
    "name": "Union Aerospace Corp",
    "address": "939 Edison Avenue",
    "city": "New York",
    "zip": "33456",
    "country": "US",
    "contactName": "T. Kelliher",
    "contactEmail": "tkelliher@uac.com",
    "telephoneNumber": "555-264862",
    "status": "ACTIVE",
    "customerId": 236,
    "groupId": 1
  }
..
```

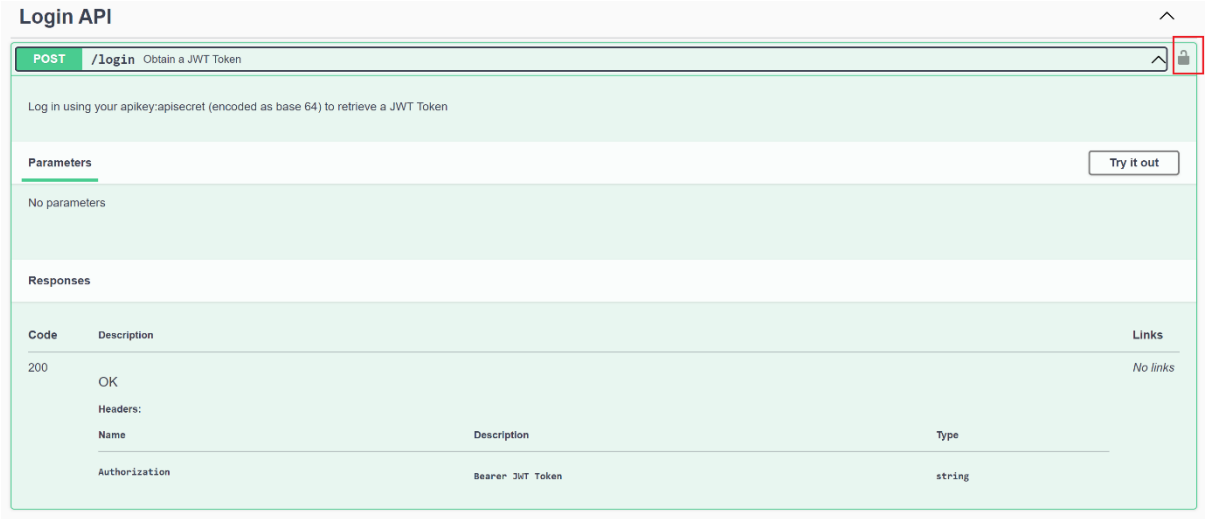

For maximum efficiency it is recommended to reuse the access token obtained by the login process until it expires, or access is no longer required, i.e. step 4 can be repeated to perform as many API calls as are necessary.

Using the Swagger UI

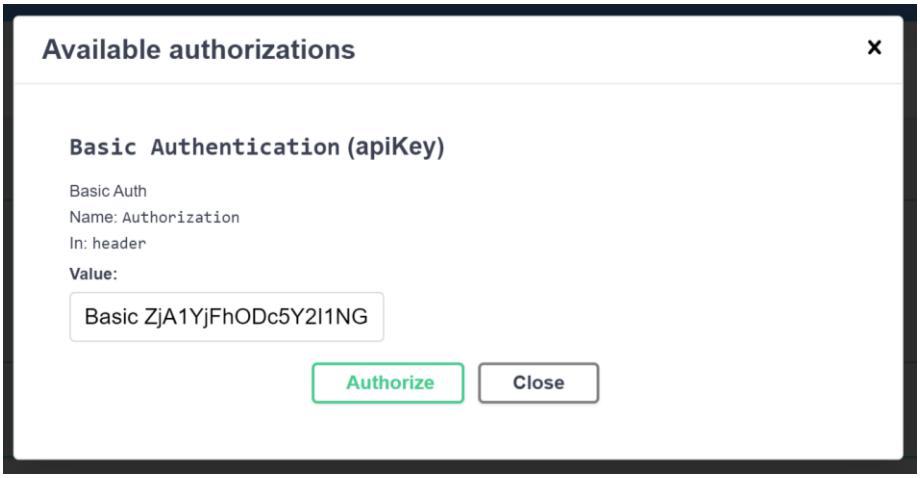
As well as providing documentation for the Insight API, the Swagger UI can be used interactively. The process for authenticating via the Swagger UI is described below:

a) Locate the 'Login API' on the Swagger page.

b) Click the 'lock' icon in the top right corner of the box as shown below.



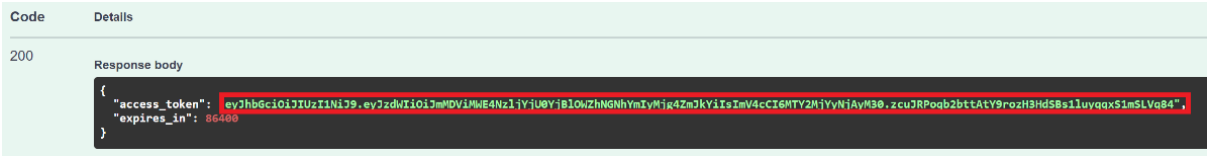
c) This will open another dialog as below whereby you can input your Basic authentication value. Using the Base64 encoded Key:Secret described above (*2. Encode the key*) enter this in the "Value:" field e.g., "Basic ZjA1YjFhODc5Y2I1NGIwZTlmYTRjYWJiMjI4OGZiZGI6RFdkTXRyVE0ySnJvQW1LczdEZkdqY2hHTGNHckpINEQ3bWxCZXVFeWJGWFZlZ3V6bkRPUmpHb0pYYldES1J2dg==”



Click the "Authorize" button and close the dialog.

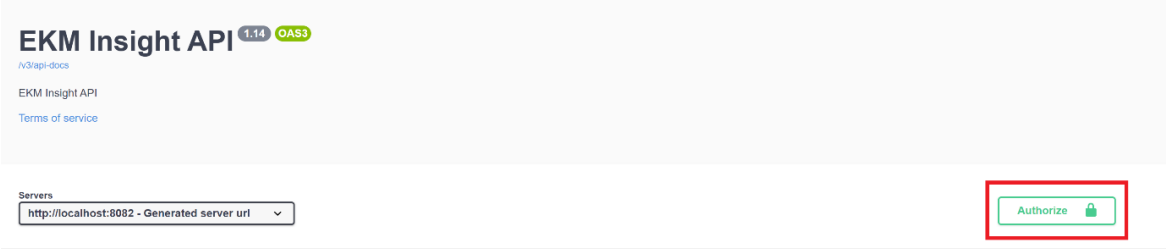
d) Use the 'Try it out' button to invoke the login API with the Authorization parameter set.

e) After clicking the 'Execute' button a response will be returned containing an authorization token or 'access_token' that looks like the following:

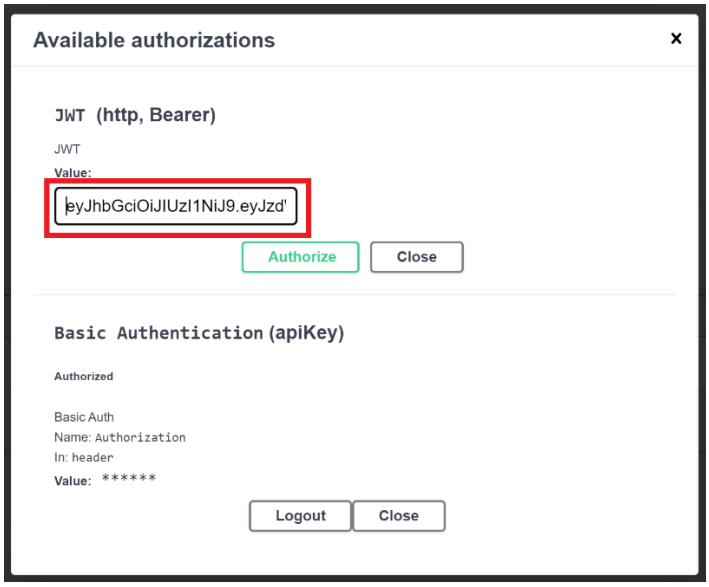


f) Copy the value of the authorization token obtained (just the unquoted text value of the access token field, not the entire JSON response).

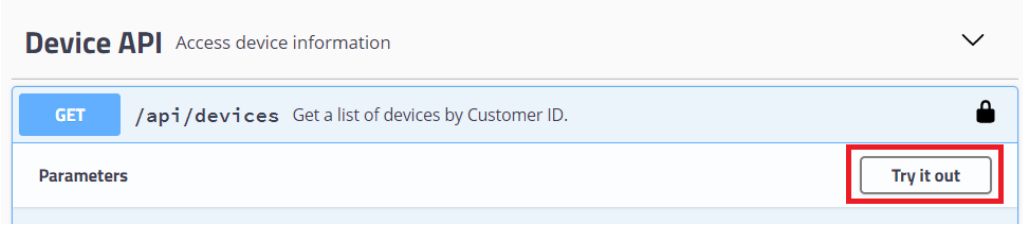
g) Click the “Authorize” button located near the very top of the page as shown below:



h) Paste your authorization token into the JWT ‘Value’ field of the authorization dialog and click the ‘Authorize’ button:



i) Other APIs can now be used interactively by clicking the ‘Try it out’ button on each API:



Appendix 1: Sample Java Insight API Client

The following code sample provides an indication of how you can create a simple Insight API client. This Java version makes use the Apache HTTP client library:

```
package com.ekm.test;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Base64;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

public class InsightApiClient {

    private static final String HOSTURL = "https://path-to-your-insight/PortalAPI/";
    private static final String LOGINURL = HOSTURL + "login";
    private static final String CUSTOMERURL = HOSTURL + "api/customers/";

    private static final String APIKEY = "077e3670b59144519de338ac141f5dec";
    private static final String APISECRET =
            "M1lklTzvaDqZNRUTG3muHxAmdnuAOf4I0KHrDwpXgsHIgChy9okwvEcX1S1td3u4";

    public static void main(String[] args) {

        //Step 2 - Base 64 encode the key and secret
        String encodedKeyAndSecret = base64encode(APIKEY, APISECRET);

        try {

            //Step 3 - Call the login interface to obtain an access token
            String jwtAccessToken = getJwtAccessToken(encodedKeyAndSecret);

            //Step 4 - Use the access token to call the Customers API
            getCustomers(jwtAccessToken);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static String base64encode(String apikey, String apisecret) {

        //concatenate the API Key + ':' + Secret
        StringBuilder sb = new StringBuilder();
        sb.append(apikey)
          .append(':')
          .append(apisecret);

        //Base 64 encode it
        return Base64.getEncoder().encodeToString(
                sb.toString().getBytes());
    }

    private static final String getJwtAccessToken(String encodedKeyAndSecret)
            throws Exception {

        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost postRequest = new HttpPost(LOGINURL);
        postRequest.addHeader("Authorization", "Basic " + encodedKeyAndSecret);
        postRequest.addHeader("accept", "application/json");

        HttpResponse response = httpClient.execute(postRequest);

        if (response.getStatusLine().getStatusCode() == 200) {

           BufferedReader br = new BufferedReader(
               new InputStreamReader((response.getEntity().getContent())));

           return extractJwtAccessToken(br.readLine());

        } else {
```

```
            throw new RuntimeException("Failed : HTTP error code : "
                    + response.getStatusLine().getStatusCode());
        }
    }

    private static void getCustomers(String jwtAccessToken)
            throws Exception {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet getRequest = new HttpGet(CUSTOMERURL);
        getRequest.addHeader("Authorization", "Bearer " + jwtAccessToken);
        getRequest.addHeader("accept", "application/json");

        HttpResponse response = httpClient.execute(getRequest);
        if (response.getStatusLine().getStatusCode() == 200) {

            BufferedReader br = new BufferedReader(
                new InputStreamReader((response.getEntity().getContent())));

            String output;
            while ((output = br.readLine()) != null) {
                System.out.println(output);
            }

         } else {
             throw new RuntimeException("Failed : HTTP error code : "
                     + response.getStatusLine().getStatusCode());
         }
    }

    private static String extractJwtAccessToken(String httpResponseBody) {
        int jwtStart = httpResponseBody.indexOf(":") + 2;
        int jwtEnd = httpResponseBody.indexOf(",", jwtStart) - 1;
        return httpResponseBody.substring(jwtStart, jwtEnd);
    }
}
```

Appendix 2: Advanced Search API

The '/api/devices/search', '/api/monitors/search' and '/api/customers/search' endpoints provide a flexible query syntax for advanced searches. The syntax for the search query parameter (q=) is:

```
field:[comparison]value [ field:[comparison]value]...
```

The 'comparison' is optional. If omitted the meaning is 'equals'. The set of comparison operators supported depends on the data type of the field:

- For numeric fields: < , >, ! (not equal to), [blank] (means equal to)
- For textual fields: ~ (matches, i.e. SQL LIKE), !~ (doesn't match), ! (is not), [blank] (is)
    - SQL wildcards %, _ and [...] are supported in ~ and !~ query terms. *Note [and] must be URI encoded to %5B and %5D in request parameters.*
    - Text search terms are not case-sensitive.
- For date & time fields: <, >

To include a space character inside a filter value term it needs to be preceded by a single backslash (\). For example: `comment:~removed\ from\ site`

Multiple field-comparison-value terms separated by a space character are combined using logical AND expressions. Alternatively, the q= parameter may be specified multiple times in the request; the result is the same.

The supported **Device** fields for searching/filtering are in the table below:

Name	Description
Textual types	
serial	The serial number
asset	Asset number (as entered at the Portal)
sticker	sticker number (as entered at the Portal)
ipAddress	The IP address
macAddress	The MAC address
hostname	The hostname if known
comment	Comment text as entered at the Portal
monitorStatus	The device monitoring status code: • Y = Fully enabled, • I = Consumables only, • J = Reports only, • X = Disabled.
monitorName	The name of the monitor that discovered and is monitoring this device location – asset location (added in API v1.9)
location	The asset location (added in API v1.9)
Date & Time Types	
lastContact	The date and time of last information received from this device
discoveryDate	The date and time the device was first discovered by the monitor
Numeric Types	
deviceId	The unique device ID
internalId	The monitor-specific local device ID
customerId	The unique ID of parent customer owning the device

Plus all custom fields except encrypted string type. To filter by a custom field specify its name with the prefix 'cf_', for example: `api/devices/search?q=cf_My_Custom_Notes:~SPECIAL`

The supported **Monitor** fields for searching/filtering are:

Name	Description
Textual types	
name	The monitor's name
remoteApplication	The type and version of the monitor, e.g. DCA7.20.2.20
Date & Time Types	
lastContact	The date and time of last information received from the monitor
Numeric Types	
monitorId	The unique ID of monitor
customerId	The unique ID of parent customer owning the monitor
Boolean	
online	Indicates whether the monitor is currently connected and reporting to the Portal
Enumeration	
status	(ACTIVE/DISCONTINUED). Discontinued monitors are "archived" and no longer in use; they may still be installed/operational but all their devices are set to Disabled.

Plus all custom fields except encrypted string type.

The supported **Customer** fields for searching/filtering are:

Name	Description
Textual types	
name	The customer's name
address	The customer's address
city	The customer's city
zip	The customer's zip or postal code
country	The customer's country
contactname	The customer contact name
contactemail	The customer contact e-mail address
telephonenumber	The customer's telephone number
Date & Time Types	
createddate	The date and time the customer was created
Numeric Types	
groupId	The unique ID of parent group owning the customer
Enumeration	
status	(ACTIVE/EXPIRED). Expired customers are "archived" and no longer in use.

Plus all custom fields except encrypted string type.

Advanced Search API Examples

Example	HTTP Operation	Example JSON Response
Find device by serial number	`GET /api/devices/search?q=serial:NL76MBCF24&includeExtendedFields=true`	`[ { "customerId": 56, "serialNumber": "VNCVF1G0DH", "monitorStatus": "Y", "assetNumber": "asset number 10", "stickerNumber": "sticker number 5", "discoveryDate": "2018-05-02T16:05:37.000Z", "lastContact": "2019-10-24T05:07:57.000Z", "deviceId": 1234, "ipAddress": "127.0.0.1" } ]`
Find devices by partial match of a serial number	`GET /api/devices/search?q=serial:~BCF24`	`[ { "customerId": 56, "serialNumber": "VNCVF1G0DH", "monitorStatus": "Y", "assetNumber": "asset number 10", "stickerNumber": "sticker number 5", "discoveryDate": "2018-05-02T16:05:37.000Z", "lastContact": "2019-10-24T05:07:57.000Z", "deviceId": 1234, "ipAddress": "127.0.0.1" } ]`

		]

Find all Fully Enabled or Consumables Only devices of customer ID 20 that have been out of contact since 15-Feb-2020:	GET /api/devices/search?
	q=customerId:20 monitorStatus:~[YI]
	lastContact:>2022-02-15
		[
		{
		"customerId": 20,
		"serialNumber": "",
		"monitorStatus": "Y",
		"assetNumber": "?",
		"stickerNumber": "?",
		"discoveryDate": "2017-02-10T09:38:04.000Z",
		"lastContact": "2022-05-15T23:20:41.000Z",
		"deviceId": 2815,
		"ipAddress": "15.13.149.90"
		},
		{
		"customerId": 20,
		"serialNumber": "CN5AO6Y005",
		"monitorStatus": "Y",
		"assetNumber": "abcde",
		"stickerNumber": "?",
		"discoveryDate": "2017-02-10T09:38:10.000Z",
		"lastContact": "2022-05-25T07:11:50.000Z",
		"deviceId": 2818,
		"ipAddress": "15.13.145.125"
		}
		]

Find active DCA monitors that have not reported data since 15-Jan-2021, and include extended data fields (contact name, email etc.) in the response:	GET
	/api/monitors/search?includeExtendedFields=true&q=status:ACTIVE lastContact:<2021-01-15
		[
		{
		"customerId": 202,
		"name": "96268",
		"remoteApplication": "DCA7.20.2.22",
		"online": false,
		"lastContact": "2020-04-06T15:11:55.000Z",
		"status": "ACTIVE",
		"extendedFields": {
		"address": "",
		"city": "",
		"zip": "",
		"country": "",
		"contactName": "",
		"contactEmail": "",
		"contactTelephone": ""
		},
		"licenceProviderCode": "",
		"licenceKey": "112233006238",
		"licenceExpiryDate": "2030-03-23",
		"licenceDeviceLimit": 1000,
		"monitorId": 278
		}
		]

Appendix 3: Device API Examples

Example	HTTP Operation	JSON Request Body Payload	JSON Response Payload

Update a device by ID	PATCH
	/api/devices/1234
		<pre>{
		  "monitorStatus": "Y",
		  "assetNumber": "asset number 10",
		  "stickerNumber": "sticker number 5",
		  "comment": "new comment 1"
		}</pre>
			<pre>{
			  "id": 1234,
			  "success": true,
			  "message": ""
			}</pre>

Update a device by ID with custom fields	PATCH
	/api/devices/1234
		<pre>{
		  "monitorStatus": "Y",
		  "assetNumber": "asset number 10",
		  "stickerNumber": "sticker number 5",
		  "comment": "new comment 1",
		  "customFieldValues": {
		  "Custom_DateTime": "2022-09-08T10:00:00",
		  "Custom_float": 123.4567
		  }
		}</pre>
			<pre>{
			  "id": 1234,
			  "success": true,
			  "message": ""
			}</pre>

Update a device by ID with custom fields that don't exist	PATCH
	/api/devices/1234
		<pre>{
		  "monitorStatus": "Y",
		  "assetNumber": "asset number 10",
		  "stickerNumber": "sticker number 5",
		  "comment": "new comment 1",
		  "customFieldValues": {
		  "Unknown_Field": "abcd" }
		}</pre>
			<pre>{
			  "id": 1234,
			  "success": true,
			  "message": "Partial success: Non existent Device custom data field name 'Unknown_Field'"
			}</pre>

Update multiple devices	PATCH /api/devices	[
		{
		"deviceId": 1234,
		"stickerNumber": "sticker number a",
		"comment": "test comment 1",
		"assetNumber": "new asset number 1",
		"monitorStatus": "Y",
		"customFieldValues" : {
		"Custom_DateTime":"2019-06-28T13:14:15",
		"Custom_float":123.4567
		}
		},
		{
		"deviceId": 4567,
		"comment": "test comment 2",
		"monitorStatus": "X",
		"customFieldValues" : {
		"Device_Location":"Manchester",
		"meterID1":"1",
		"monthly_colour_target":"70"
		}
		}
		]
			[
			{
			"id": 1234,
			"success": **true**,
			"message": "Partial success: Non existent Device custom data field name 'Custom_DateTime'; Non existent Device custom data field name 'Custom_float'"
			},
			{
			"id": 4567,
			"success": **true**,
			"message": ""
			}
			]