

# HWDB

David Demuth, Urbas Ekka, Hajime Muramatsu, Vladimir Podstavkov, Alex Wagner

1. Locations
2. Item Status
3. GET Image IDs from each of Test-history entries
4. New Item-filter menu
  - a. Status
  - b. Location
  - c. Manufacturer
  - d. Country of Origin
  - e. Item Specifications
  - f. Test Specifications
5. Readiness of the production version
6. Preparation for the 2nd HWDB tutorials

Same as 4 weeks ago...

# GET/POST Locations!

**DUNE** Hardware DB  
DEEP UNDERGROUND NEUTRINO EXPERIMENT

Home Edit Item Z00100300030-00103

Component Types **QR** **LOCATION LOG** SPECS LOG STRUCTURE LOG CONTAINER LOG TEST LOG IMAGES

Items **Component Type** Z.Sandbox.HWDBUnitTest.jabberwock

Institutions **Part ID** Z00100300030-00103-US186

Images **Location** University of Mississippi

Manufacturers **Serial Number** N2C7D345C

Users **Country of Origin** United States

Admin < **Resp. Institution** University of Minnesota Twin Cities

Logout **Manufacturer** Hajime Inc

**The latest location of this Item. Not editable... here**

**Comments** Here are some comments

**Component Status** temporarily not available

**Contained in** N/A

**Specifications** +

**Sub-components** +

SAVE DONE

Same as 4 weeks ago...

# GET/POST Locations!

**DUNE Hardware DB**  
DEEP UNDERGROUND NEUTRINO EXPERIMENT

Location history for [Z00100300030-00103](#)

Home

Component Types **ADD NEW...**

Items

Institutions

Images

Manufacturers

Users

Admin <

Logout

Requests, Issues?

Created	Creator	Comments	Arrival_date	Location
2024-04-10 14:09:44.316278-05:00	<a href="#">Hajime Muramatsu</a>	now testing through the REST API	2024-04-10 14:50:12-05:00	<a href="#">University of Mississippi</a>
2024-04-10 09:51:44.735194-05:00	<a href="#">Hajime Muramatsu</a>	testing...	2024-04-10 09:51:44.735194-05:00	<a href="#">University of Minnesota Twin Cities</a>
2024-04-04 08:16:44.333658-05:00	<a href="#">Hajime Muramatsu</a>	now testing through the REST API	2024-04-10 14:50:12-05:00	<a href="#">Universidade Federal de Alfnas</a>
2024-04-04 08:16:31.867400-05:00	<a href="#">Hajime Muramatsu</a>	now testing through the REST API	2024-04-06 14:50:12-05:00	<a href="#">Universidade Estadual de Campinas</a>
2024-04-04 08:14:11.847348-05:00	<a href="#">Hajime Muramatsu</a>	now testing through the REST API	2024-04-05 14:50:12-05:00	<a href="#">University of Minnesota Twin Cities</a>
2024-04-04 07:51:10.808623-05:00	<a href="#">Hajime Muramatsu</a>	testing	2024-04-04 08:56:59-05:00	<a href="#">University of Arizona</a>
2024-04-04 07:48:26.051054-05:00	<a href="#">Hajime Muramatsu</a>	testing...	2024-04-03 12:00:19-05:00	<a href="#">CERN</a>

- You can edit... ADD entries here.

CanNOT delete an entry.

- The "Arrival date", specified with a time-zone (default = CST; see the next page)

Same as 4 weeks ago...

# GET/POST Locations!

Edit Location for Item Z00100300030-00103

Location

--SELECT--

Created by

Hajime Muramatsu

Comments

Arrival Date

2024-04-15 16:27:16-05:00

SAVE

- The Arrival Date box is pre-populated with the **current** Chicago time.
- So unless you need to specify a time other than current time, no need to touch it.
- And of course, if you need to specify a different time, you can.

Same as 4 weeks ago...

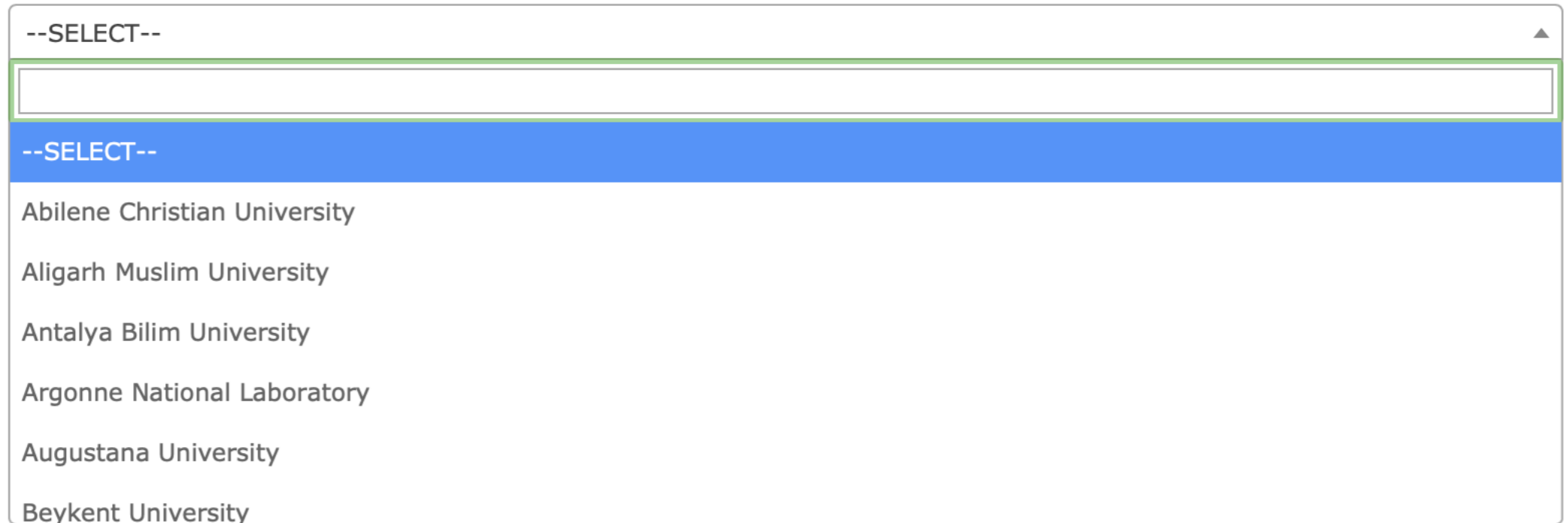
# GET/POST Locations!

Location

Created by

Comments

Arrival Date



The screenshot shows a web form with a dropdown menu for selecting a location. The dropdown is currently open, displaying a list of institutions. The top option is "--SELECT--". Below it, the following institutions are listed: Abilene Christian University, Aligarh Muslim University, Antalya Bilim University, Argonne National Laboratory, Augustana University, and Beykent University. The dropdown menu has a blue header bar with "--SELECT--" and a small upward-pointing triangle on the right side.

- You don't type in locations.
- Select your location from the pre-defined "Institution list".
- Right, if your Item location needs to change like;  
from U of Minnesota Lab Build. 8 West side shed  
to U of Minnesota the Lab Build. 8 East side shed  
these two location names need to be defined in the Institution list prior to its usage.
- Only those with the **Admin.** privilege can modify the Institution list.

# GET/POST Locations - possible issue

- Right, if your Item location needs to change like;
  - from U of Minnesota Lab Build. 8 West side shed
  - to U of Minnesota the Lab Build. 8 East side shedthese two location names need to be defined in the Institution list prior to its usage.
- So.. if you have 10 lab rooms (or buildings.. etc), there could be 10 new “institution names” associated with U of Minnesota.
- We currently have 208 institutions defined in the dev. HWDB.
- We do NOT want to have ~2000 (or more) institutions.  
(that could also break the PID syntax: e.g., Z00100300030-00168-US186)
- Suggestion:
  - ▶ Define the default (main) Institution names first in the pro. version.  
This seems to have been done already.
  - ▶ To allocate some space for future new members,  
**Why not start these “Location Institutions” from Institution ID of 1000?**

Almost same as 4 weeks ago...

# GET/POST Locations!

/api/v1/components/<PID>/locations

```

{
  "data": [
    {
      "arrived": "2024-04-10T14:50:12-05:00",
      "comments": "now testing through the REST API",
      "created": "2024-04-10T14:09:44.316278-05:00",
      "creator": "Hajime Muramatsu",
      "id": 92,
      "link": {
        "href": "/cdbdev/api/v1/locations/92",
        "rel": "details"
      },
      "location": {
        "id": 187,
        "name": "University of Mississippi"
      }
    },
    {
      "arrived": "2024-04-10T09:51:44.735194-05:00",
      "comments": "testing...",
      "created": "2024-04-10T09:51:44.735194-05:00",
      "creator": "Hajime Muramatsu",
      "id": 91,
      "link": {
        "href": "/cdbdev/api/v1/locations/91",
        "rel": "details"
      },
      "location": {
        "id": 186,
        "name": "University of Minnesota Twin Cities"
      }
    },
    {
      "arrived": "2024-04-10T14:50:12-05:00",
      "comments": "now testing through the REST API",
      "created": "2024-04-04T08:16:44.333658-05:00",
      "creator": "Hajime Muramatsu",
      "id": 15,

```

GET

- and you can do the same through the REST API.
- GET & POST
- For the Arrival Date, you could omit the time-zone info, if the provided time is already in the CST zone.

```

{
  "arrived": "2024-04-10 14:50:12",
  "comments": "now testing through the REST API",
  "location": {
    "id": 187
  }
}

```

POST

Same as 4 weeks ago...

# What else are new?

Much cleaner side-menu!

NEUTRINO EXPERIMENT

Edit Item Z00100300030-00103

Home

Component Types

Items

Institutions

Images

Manufacturers

Users

Admin <

Logout

Requests, Issues? ↗

QR LOCATION LOG SPECS LOG STRUCTURE LOG CONTAINER LOG TEST LOG IMAGES

Component Type Z.Sandbox.HWDBUnitTest.jabberwock

Part ID Z00100300030-00103-US186

Location University of Mississippi

Serial Number SN2C7D345C

Country of Origin United States

Resp. Institution University of Minnesota Twin Cities

Manufacturer Hajime Inc

Created 2024-04-03 12:00:19

Created by Hajime Muramatsu

Specs Version

Comments Here are some comments

Component Status temporarily not available

Contained in N/A

Specifications +

Sub-component... +

SAVE DONE

Component Status!



Almost same as 4 weeks ago...

# Component Status

The screenshot shows a web interface for 'Component Status'. On the left is a sidebar with five tabs: 'Comments', 'Component Status', 'Contained in', 'Specifications', and 'Sub-components'. The 'Component Status' tab is active. The main content area contains a text input field with the placeholder 'Here are some comments'. Below it is a dropdown menu with four options: 'temporarily not available', 'available', 'temporarily not available', and 'permanently not available'. The 'available' option is highlighted with a blue background. At the bottom of the form, there are two buttons: a blue one and a white one.

- **3 choices:**
  - available
  - temporarily not available
  - permanently not available
- **When you add a new Item, the initial status is available.**
- **When you make sub-component links, the employed Items must be in available.**

# GET/PATCH Status!

`/api/v1/components/<PID>/status`

**GET**

```
{
  "component": {
    "id": 152958,
    "part_id": "D00599800001-00379"
  },
  "data": {
    "status": {
      "id": 3,
      "name": "permanently not available"
    }
  },
  "link": {
    "href": "/cdbdev/api/v1/components/D00599800001-00379/status",
    "rel": "self"
  },
  "status": "OK"
}
```

**PATCH**

```
{
  "component": {
    "part_id": "D00599800001-00379"
  },
  "status": {
    "id": 1
  }
}
```

- Specify Status by an integer:
  - 1 = available
  - 2 = temporarily not available
  - 3 = permanently not available

# Image IDs of each Test entries

```
"data": [  
  {  
    "comments": "here is the 1st test of the test result...",  
    "created": "2020-11-16T10:13:48.704421-06:00",  
    "creator": {  
      "id": 12624,  
      "name": "Hajime Muramatsu",  
      "username": "hajime3"  
    },  
    "id": 110,  
    "images": [  
      {  
        "id": "12529424-86a6-11eb-a31b-936fd91bb429",  
        "name": "DFD-20-A017.pdf"  
      }  
    ],  
    "link": {  
      "href": "/cdbdev/api/v1/component-tests/110",  
      "rel": "details"  
    },  
    "methods": [  
      {  
        "href": "/cdbdev/api/v1/component-tests/110/images",  
        "rel": "Images"  
      }  
    ],  
    "test_data": {  
      "Cleaned": 0,  
      "Comments": null,  
      "Template": 1,  
      "Visual": 40  
    },  
    "test_spec_version": -1,  
    "test_type": {  
      "id": 15,  
      "name": "CPA_Parts_FR4_main QC check"  
    }  
  }  
],
```

- (with the WEB UI, this procedure is straightforward.. one wouldn't need to know the corresponding ID)
- Now, Image ID (along with OID ←— see our tutorials...) is reported when you GET a Test result. This provides an easy access to images that are stored in each Test entries.
- You can access to images that are associated with older Test entries as well.
- Image ID = 12529424-86a6-11eb-a31b-936fd91bb429  
With this, one can retrieve its stored image.
- OID = 110
- Image Name = DFD-20-A017.pdf

## **New fields in the Item filters**

**These allow us to filter Items (components) more thoroughly  
and create/dump a list of Items**

# New field in the Item filters

## - Status -

Apply filters

Component\_type

Part\_id

Serial\_number

Manufacturer

Creator

Comments

Location

Country\_of\_origin

Resp\_institution

Status --any--

Specifications

Test\_data

SEARCH CANCEL

Apply filters

Component\_type

Part\_id

Serial\_number

Manufacturer

Creator

Comments

Location

Country\_of\_origin

Resp\_institution

Status

Specifications

Test\_data

SEARCH CANCEL

- ✓ --any--
- available
- temporarily not available
- permanently not available

This will display a list of PIDs that have a specific “Status”.

“any” means any of the 3 choices (i.e., all of them).

# New field in the Item filters

## - Location -

Can filter the latest location as well!

Apply filters

Component_type	<input type="text"/>
Part_id	<input type="text"/>
Serial_number	<input type="text"/>
Manufacturer	<input type="text"/>
Creator	<input type="text"/>
Comments	<input type="text"/>
Location	<input type="text" value="minn"/>
Country_of_origin	<input type="text"/>
Resp_institution	<input type="text"/>
Status	<input type="text" value="--any--"/>
Specifications	<input type="text"/>
Test_data	<input type="text"/>

- No need to completely spell out.
- Case insensitive.

# New field in the Item filters

## - Manufacturer -

Apply filters

Component_type	<input type="text"/>
Part_id	<input type="text"/>
Serial_number	<input type="text"/>
Manufacturer	<input type="text" value="cERn"/>
Creator	<input type="text"/>
Comments	<input type="text"/>
Location	<input type="text"/>
Country_of_origin	<input type="text"/>
Resp_institution	<input type="text"/>
Status	<input type="text" value="--any--"/>
Specifications	<input type="text"/>
Test_data	<input type="text"/>

- No need to completely spell out.
- Case insensitive.

# Filtering Items through the REST API

- Manufacturer -

`/api/v1/components?manufacturer=<XXX>`

**GET**

E.g.,

`/api/v1/components?manufacturer=hajime%20inc" | jq .data[].part_id`

- Case insensitive.



# New field in the Item filters

## - Country of Origin -

The image shows a screenshot of a web application's 'Apply filters' dialog box. The dialog is titled 'Apply filters' and contains a list of filter fields on the left and their corresponding input areas on the right. The fields are: Component\_type, Part\_id, Serial\_number, Manufacturer, Creator, Comments, Location, Country\_of\_origin, Resp\_institution, Status, Specifications, and Test\_data. The 'Country\_of\_origin' field is highlighted with a red dashed border and contains the text 'japan'. The 'Status' field is a dropdown menu with the text '--any--'. At the bottom right of the dialog, there are two buttons: 'SEARCH' and 'CANCEL'.

Field Name	Value
Component_type	
Part_id	
Serial_number	
Manufacturer	
Creator	
Comments	
Location	
Country_of_origin	japan
Resp_institution	
Status	--any--
Specifications	
Test_data	

- No need to completely spell out.
- Case insensitive.

# New field in the Item filters

## - Item Specifications -

Suppose I have the following in one of the Items (D00400300001-00340)

```
"specifications": [  
  {  
    "Documentation": "https://something.com/whatever",  
    "SiPM_Strip_ID": 4548,  
    "Test_Box_ID": "M105",  
    "Tray_Number": 20,  
    "Vendor_Box_Number": 14,  
    "Vendor_Delivery_ID": "HPK_Ciemat_03",  
    "_meta": {  
      "_column_order": [  
        "Vendor_Delivery_ID",  
        "Vendor_Box_Number",  
        "Tray_Number",  
        "SiPM_Strip_ID",  
        "Test_Box_ID",  
        "Documentation"  
      ]  
    }  
  }  
],
```

and I want to dump a list of PIDs that have SiPM\_Strip\_ID = 4548.

# New field in the Item filters

## - Item Specifications -

### Apply filters

<b>Component_type</b>	<input type="text"/>
<b>Part_id</b>	<input type="text"/>
<b>Serial_number</b>	<input type="text"/>
<b>Manufacturer</b>	<input type="text"/>
<b>Creator</b>	<input type="text"/>
<b>Comments</b>	<input type="text"/>
<b>Status</b>	available <input type="button" value="v"/>
<b>Specifications</b>	SiPM_Strip_ID==4548
<b>Test_data</b>	<input type="text"/>

**“SiPM\_Strip\_ID==4548” does the job**

# New field in the Item filters

## - Item Specifications -

Suppose I have the following in one of the Items (D00400300001-00340)

```
"specifications": [  
  {  
    "Documentation": "https://something.com/whatever",  
    "SiPM_Strip_ID": 4548,  
    "Test_Box_ID": "Mib3",  
    "Tray_Number": 20,  
    "Vendor_Box_Number": 14,  
    "Vendor_Delivery_ID": "HPK_Ciemat_03",  
    "_meta": {  
      "_column_order": [  
        "Vendor_Delivery_ID",  
        "Vendor_Box_Number",  
        "Tray_Number",  
        "SiPM_Strip_ID",  
        "Test_Box_ID",  
        "Documentation"  
      ]  
    }  
  }  
],
```

and I want to dump a list of PIDs that have Test\_Box\_ID = Mib3.

# New field in the Item filters

## - Item Specifications -

### Apply filters

<b>Component_type</b>	<input type="text"/>
<b>Part_id</b>	<input type="text"/>
<b>Serial_number</b>	<input type="text"/>
<b>Manufacturer</b>	<input type="text"/>
<b>Creator</b>	<input type="text"/>
<b>Comments</b>	<input type="text"/>
<b>Status</b>	available <input type="button" value="v"/>
<b>Specifications</b>	Test_Box_ID==Mib3
<b>Test_data</b>	<input type="text"/>

**“Test\_Box\_ID==Mib3” does the job**

# Syntax in the filtering field

## - For numbers:

- ▶ ==
- ▶ !=
- ▶ <
- ▶ <=
- ▶ >
- ▶ >=

## - For strings/substrings:

- ▶ ==
- ▶ !=
- ▶ ~ (regex search : case sensitive) : See later pages
- ▶ ~\* (regex search : case insensitive) : See later pages

# Syntax in the filtering field

- regex -

- Suppose I have the following in one of the Items (D00400300001-00340):

```
"specifications": [  
  {  
    "Documentation": "https://something.com/whatever",  
    "SiPM_Strip_ID": 4548,  
    "Test_Box_ID": "Mib3",  
    "Tray_Number": 20,  
    "Vendor_Box_Number": 14,  
    "Vendor_Delivery_ID": "HPK_Ciemat_03",  
    "_meta": {  
      "_column_order": [  
        "Vendor_Delivery_ID",  
        "Vendor_Box_Number",  
        "Tray_Number",  
        "SiPM_Strip_ID",  
        "Test_Box_ID",  
        "Documentation"  
      ]  
    }  
  }  
],
```

- case sensitive : Vendor\_Delivery\_ID~HPK\_Ciemat\_
- case sensitive : Vendor\_Delivery\_ID~HPK\_CiemaT\_ : Nothing returned.
- case insensitive : Vendor\_Delivery\_ID~\*HPK\_CiemaT\_

# Syntax in the filtering field

- nested structure -

- Suppose I have the following in one of the Items (D00599800001-00379):

```
"specifications": [  
  {  
    "DATA": {  
      "Drawing Number": "DFD-21-2101",  
      "Label Code": "12345",  
      "Name": "Main I-Beam"  
    }  
  }  
]
```

You can still filter it simply via;

**Name==Main I-Beam**

or

**Label Code==12345**



# Filtering Items through the REST API

- Item Specification -

`/api/v1/components?specs=<XXX>`

**GET**

E.g.,

`/api/v1/components?specs=SiPM_Strip_ID==4548 | jq .data[].part_id`

`/api/v1/components?specs=Test_Box_ID==Mib3 | jq .data[].part_id`

`/api/v1/components?specs=Name==Main%20I-Beam | jq .data[].part_id`

`/api/v1/components?specs=Label%20Code==12345 | jq .data[].part_id`

# New field in the Item filters

## - Test Specifications -

Apply filters

**Component\_type**

**Part\_id**

**Serial\_number**

**Manufacturer**

**Creator**

**Comments**

**Status** available ▾

**Specifications**

**Test\_data**

SEARCH CANCEL

**We can apply a filter to Test Specifications in the same way!**

# Filtering each element of an array

- Suppose I have the following in one of the Items (D00400300001-00380):

```
"test_data": {  
  "Test Results": [  
    {  
      "Date": "01-26-2024-06:13 UTC",  
      "Location": "Milano Bicocca",  
      "Operator": "Andrea Falcone; Claudia Brizzolari; Francesco Terranova; Luca Meazza; Maritza Delgado; Maurizio Perego",  
      "Polarization": "Rev",  
      "SiPM": [  
        {  
          "Comment": "",  
          "I": [  
            1.2604860486048603e-06,  
            7.759315931593157e-07,  
            1.2117011701170138e-07,  
            8.730783078307831e-07,  
            1.4496849684968533e-07,  
            4.010081008100806e-07,  
            5.816921692169214e-07,  
            2.153726372637264e-06,  
            1.612241224122412e-06,  
            -6.832583258325834e-07
```

The followings also work:

- $I[0]==1.2604860486048603e-06$  : the 1st index
- $I[0]>1.260486e-06$  : the 1st index
- $I[1]==7.759315931593157e-07$  : the 2nd index
- $I[*]==1.2117011701170138e-07$  : Any of them

Here, the right-hand sides happen to be numbers. But this works for strings as well.

# Filtering Items through the REST API

- Test Specification -

`/api/v1/components?testdata=<XXX>`

**GET**

E.g.,

`/api/v1/components?testdata=RPFSSRes%20RP%20Drawing%20N==DFD-20-A503 | jq`

`/api/v1/components?testdata=l[0]==1.2604860486048603e-06 | jq .data[].part_id`

By the way, if `[]` were disliked in your terminal,  
try to replace `[` and `]` by `\[` and `\]`.  
(or even by `%5B` and `%5D`)

# Summary of Filtering Items through the REST API

```
/api/v1/components?part_type_id=<XXX>  
/api/v1/components?part_ids=<XXX>  
/api/v1/components?serial_number=<XXX>  
/api/v1/components?manufacturer =<XXX>  
/api/v1/components?creator=<XXX>  
/api/v1/components?comments =<XXX>  
/api/v1/components?specs=<XXX>  
/api/v1/components?testdata=<XXX>
```

# The production version of the HWDB

**Thanks to Vladimir to always promptly response to our requests!**

**With the updates we described today,  
we believe the production version is ready!!**

**.... ok, ALMOST ready.**

**We like to check the new filtering functionality some more.**

**When we are done with the tests,**

**we'll let Norm/Ana Paula know. It will be in one or two days.**

**The 2nd HWDB Tutorials  
&  
The DUNE HWDB Training site**



# The 2nd HWDB Tutorials

- **When** : in roughly 3 weeks.

We like to finalize the Github-IO site,  
but also need time to come up with examples, including DB schema.

Thanks to David for setting up the Github-IO site and  
teaching us how-to create pages!

It will be 2-day tutorials, just like 2 years ago.

- **Where** : Online, just like 2 years ago.

# Tentative schedule

## Day 1:

- 30 min : Overview of HWDB
- 30 min : Intro. to the PID
- 30 min : Setting up Component Type Definitions

## Day 2:

- 15 min : Common questions/issues from Day 1
- 30 min : Entering/retrieving data through WEB UI
- 30 min : Entering/retrieving data through REST API
- 15 min : Entering/retrieving data through apps

2022

## Day 1:

- 30 min : Intro. to the PID and setting up Types
- 45 min : Entering/retrieving data through WEB UI
- 45 min : Entering/retrieving data through REST API

## Day 2:

- 10 min : A quick overview of Day 1
- 20 min : the iPad app : QR codes, Locations, and PID hierarchy!
- 60 min : **Uploading massive data via Python API (might split into 2 sub-sections)**

2024

Like to emphasize on the real/practical examples,  
with the ready-to-be-used examples (i.e., excel sheets with real DB schema)

# The DUNE HWDB Training Site

- The setup page is there now...

Home Code of Conduct Setup Episodes Extras License Improve this page

Search...

## DUNE HWDB Training: Obtaining a FNAL certificate

### Objectives

- Get ready to do the tutorials
- Obtain a certificate to access to the HWDB (both development and production version)

### Requirement

- You must have a FNAL Services account and be on the DUNE Collaboration member list.
- You also need to have your terminal ready.
- During our "REST API" sessions, we will extensively use the command, cURL (client URL, pronounced "curl", <https://curl.se/>).

### Obtaining your certificate

By following the procedure described below, you will obtain your password-protected certificate in the "PKCS #12 format" that bundles a private key with its X.509 certificate from <https://www.cilogon.org>.

1. With your web browser, go to <https://www.cilogon.org>
2. Select **Log On** from its side menu.
3. Select **Fermi National Accelerator Laboratory** as your Identity Provider and click **Log On**.
4. Provide your FNAL Services account credential.
5. Select **Create Password-Protected Certificate** (usually shows up at the top).
6. Enter the password, which is not necessarily the same as the one for your FNAL Services account (preferentially a different password).
7. Select **Get New Certificate**.
8. Select **Download Your Certificate**.
9. Select **Log Off** (or close your browser).
10. You should have a file, **usercred.p12**, downloaded on your computer now. Move it to an appropriate area. We also recommend to rename the file.

### Let's try to use it!

This exercise will use your downloaded certificate to communicate with the REST API of the HWDB.

1. This step may not be necessary. But there are certain lines that repeatedly show up in curl commands during the tutorials. So let us define the followings:

**Code**

```
alias CURL='curl --cert-type P12 --cert usercred.p12:PassWord'
export APIPATH='https://dbwebapi2.fnal.gov:8443/cdbdev/api/v1'
```

In the above, **usercred.p12** is your downloaded certificate. We are assuming it sits in the current directory. Else, provide the appropriate path in front of it.

**PassWord** is the one you provided in the Step 6 in the above procedure.

**cdbdev** allows us to communicate with the development version of the HWDB.

2. Let's use an API endpoint, **/users/whoami**, to display your HWDB account info.

**Bash**

```
CURL "${APIPATH}/users/whoami"
```

📌 If everything is correct, you should see your account information in JSON like the following:

**Output**

```
{"data":{"active":true,"administrator":true,"affiliation":"University of Minnesota","architect":true,"email":"hramat@umn.edu","full_name":"Hajime Muramatsu","roles":[{"id":30,"name":"HVS-CPA"}, {"id":32,"name":"HVS-EW"}, {"id":31,"name":"HVS-FC"}, {"id":4,"name":"tester"}, {"id":3,"name":"type-manager"}],"user_id":12624,"username":"hajime3"},"link":{"href":"/cdbdev/api/v1/users/12624","rel":"self"},"status":"OK"}
```

Sometimes the response might be too long. If commands like json\_pp or jq are available, you could also pipe into them:

**Bash**

```
CURL "${APIPATH}/users/whoami" | json_pp -json_opt pretty,canonical
or
CURL "${APIPATH}/users/whoami" | jq
```

Then the above JSON response would look nicer, easier to read as the following:

**Output**

```
{
  "data": {
    "active": true,
    "administrator": true,
    "affiliation": "University of Minnesota",
    "architect": true,
    "email": "hramat@umn.edu",
    "full_name": "Hajime Muramatsu",
    "roles": [
      {
        "id": 30,
        "name": "HVS-CPA"
      },
      {
        "id": 32,
        "name": "HVS-EW"
      },
      {
        "id": 31,
        "name": "HVS-FC"
      },
      {
        "id": 4,
        "name": "tester"
      },
      {
        "id": 3,
        "name": "type-manager"
      }
    ],
    "user_id": 12624,
    "username": "hajime3"
  },
  "link": {
    "href": "/cdbdev/api/v1/users/12624",
    "rel": "self"
  },
  "status": "OK"
}
```

DUNE HWDB Training

# Setting up Types

## Overview

Teaching: 30 min  
Exercises: 0 min

### Questions

- How do we define a Component Type and a Test Type with the WEB UI?
- How about through the REST API?

### Objectives

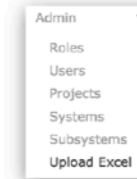
- Users should be able to setup both Component Types and Test Types so that they would be ready to enter actual data.
- In setting up a Component Type, users should be also able to define lists of **Roles**, **Manufacturers**, and **Connectors** (Type IDs of sub-components).

## Contents

	Description
<b>WEB UI</b>	
<a href="#">Going through the hierarchy of IDs</a>	List of Projects, Systems, Subsystems, and Component Types
<a href="#">Defining a Component Type</a>	Definition of a Component Type
<a href="#">Managed by</a>	Permissions of editing a Component Type definition
<a href="#">Manufacturers</a>	List of Manufacturers, creating and editing
<a href="#">Specifications</a>	DB schema for a Component Type
<a href="#">Connectors</a>	where sub-Component Types are defined
<a href="#">SPEC LOG</a>	History of Specifications
<a href="#">Defining a Test Type</a>	Definition of a Test Type
<b>REST API</b>	
<a href="#">Going through the hierarchy of IDs with the REST API</a>	List of Projects, Systems, Subsystems, and Component Types
<a href="#">GET a list of Projects</a>	GET /projects
<a href="#">GET lists of Systems, Subsystems, and Component Types</a>	GET /systems, /subsystems, and /component-types
<a href="#">Defining a Component Type with the REST API</a>	PATCH & GET /component-types/<type_id>
<a href="#">Defining a Test Type with the REST API</a>	POST & GET /component-types/<type_id>/test-types
<a href="#">Some other useful endpoints</a>	List of some useful REST API endpoints

### Going through the hierarchy of IDs

In the previous session, we introduced a several different types of IDs: **Project**, **System ID**, **Subsystem ID**, and **Component Type ID**. With the WEB UI, one can go down (and up) such hierarchy of IDs easily. Click **Admin** in the side-menu. It will show a sub-menu as shown here. One can select any of the three, **Projects**, **Systems**, or **Subsystems** to get into each of those lists.



Let's start with the root by clicking **Projects** in the above sub-menu. You should be seeing a list of currently available Projects in the HWDB like this:

Project_id	Project_name	Systems	Creator	Created	Comments
a	Dummy a		Vladimir Podstavkov	2021-10-28 10:14:44UTC-05:00	None
Z	Sandbox		Vladimir Podstavkov	2022-01-28 16:22:22UTC-06:00	Test project
D	DUNE		Vladimir Podstavkov	2022-01-31 16:12:30UTC-06:00	Includes approved far detector modules and near detectors

Now let's click the folder in the **DUNE** row & **Systems** column. This will show a list of **Systems** (individual **System** names along with **System IDs**) that correspond to the **DUNE** project as shown below:

Project_id	System_id	System_name	Subsystems	Creator	Created	Comments
D	Invalid			Vladimir Podstavkov	2022-02-01 17:24:12UTC-06:00	
D	1	FD L-HD Complete Detector		Vladimir Podstavkov	2022-02-01 17:24:12UTC-06:00	
D	2	FD L-HD Instrumented Arched Plane (with Elec and photon Det.)		Vladimir Podstavkov	2022-02-01 17:24:12UTC-06:00	

### GET lists of Systems, Subsystems, and Component Types

Similarly one can easily *get* lists of Systems, Subsystems, and Component Types with the following endpoints:

- List of Systems: /systems/<project\_id>
- List of Subsystems: /subsystems/<project\_id>/<system\_id>
- List of Component Types: /component-types/<project\_id>/<system\_id>/<subsystem\_id>

### Quiz

GET lists of Systems, Subsystems, and Component Types for the following cases:

- Obtain a list of Systems for project\_id = D
- Obtain a list of Subsystems for project\_id = D and system\_id = 005
- Obtain a list of Component Types for project\_id = D, system\_id = 005, and subsystem\_id = 20

[back to top](#)

### Defining a Component Type with the REST API

Now let's define a Component Type. Let's take the **Front Axle** (Type ID = Z00100400005) as an example.

The endpoint we need in this case is /component-types/<type\_id> Or the actual command-line would look like the following:

```
Bash
CURL -H "Content-Type: application/json" -X PATCH -d @Patch_CompType.json "${APIPATH}/component-types/Z00100400005" | jq
```

Here:

- we are *patching* instead of *posting* as the Component Type must pre-exist (only **Architect** can create a Component Type)
- Type ID = Z00100400005
- Patch\_CompType.json is a file and its contents are shown below:

```
Code
{
  "part_type_id": "Z00100400005",
  "comments": "Setting up this Type",
  "manufacturers": [7,27],
  "roles": [3,4],
  "properties": {
    "specifications": {
      "last name": "Muramatsu",

```

- "Setting up Component&Test Types" page is also there....

## The DUNE HWDB Training site

- This week, we are now updating the tutorial talks, especially to include these latest changes on the DB side.
- Next week, we'll start to implement these updated materials to the HWDB training site.  
This should be done in two weeks.
- ... so that the site should be ready by the time we have the 2nd tutorials, along with the production version.