

Component Types

**Let's prepare to Setup the Component Types in the HWDB
(we'll enter hardwares in DAY 2)**

**I will go through this session, first with the WEB UI,
and then with the REST API.**

ID...

- We just heard Jim's talk on PID. If you haven't, read the PID DOC:

<https://edms.cern.ch/document/2505353/3>

- Again, every DUNE part has an identifier, this is its PID.

D/I/L/P	001-999	001-999	00001-99999	-	00001-99999	-	AA-ZZ	001-999	-	00-99	00-99	001-999
Project	System ID	Subsystem ID	Component Type ID	Dash	Item Number	Dash	Country of Origin	Responsible Institution ID	Dash	Detector ID	<u>Final Destination</u>	Intermediate Destination
F	F	F	F		F		F	F		M	M	M

- And every DUNE part is referred to as an **Item** in HWDB.

And to communicate with HWDB,
we'll have to deal with different kinds of IDs.

type id, eid, and cid

- Every Item in HWDB is a certain Type of Component and the full component type id or type id is coded into the first 4 fields of the PID:

D/I/L/P	001-999	001-999	00001-99999
Project	System ID	Subsystem ID	Component Type ID

- Every Item is the n th example of that Type, the n is coded into the 5th field. The first 5 fields of the PID provide a unique reference to an Item which HWDB calls the External ID or eid.

D/I/L/P	001-999	001-999	00001-99999	-	00001-99999
Project	System ID	Subsystem ID	Component Type ID	Dash	Item Number

- Finally, when you create an Item then you need to complete 2 other mandatory fields of the PID, Country of Origin and Responsible Institution ID. Call this as cid.

D/I/L/P	001-999	001-999	00001-99999	-	00001-99999	-	AA-ZZ	001-999
Project	System ID	Subsystem ID	Component Type ID	Dash	Item Number	Dash	Country of Origin	Responsible Institution ID

... if you feel lost, that is ok. Just remember that;

- **type id**: The first 4 fields of the PID:

D/I/L/P	001-999	001-999	00001-99999
Project	System ID	Subsystem ID	Component Type ID

- **eid** : The first 5 fields of the PID:

D/I/L/P	001-999	001-999	00001-99999	-	00001-99999
Project	System ID	Subsystem ID	Component Type ID	Dash	Item Number

- **cid** : The first 7 fields of the PID:

D/I/L/P	001-999	001-999	00001-99999	-	00001-99999	-	AA-ZZ	001-999
Project	System ID	Subsystem ID	Component Type ID	Dash	Item Number	Dash	Country of Origin	Responsible Institution ID

We often refer to type id, eid, and cid

Don't confuse them with PID.

What is Component Type?

- In HWDB, every entry is an **Item**.

E.g., 50 CPA FR4 frame main support bars are **Items**.

- A **Component Type** is like a commercial model number and an **Item** is its serial number.

It defines what type of Items is being referenced.

E.g., CPA FR4 frame main support bar is a **Component Type**.

- Before creating Items one needs to define what is the data to be associated with each Items.

- Data can be information and/or relation to other components.

Examples of Component Types

- **Information:** Anything that is associated with its Items.

E.g., their manufacturer(s), drawing number

- **Relation:** Connections to different Component Type(s).

Useful when one needs to link between a parent and a daughter Item(s).

E.g., an assembly is a Component Type that is constructed from other Component Types (A CPA Plane is made of CPA Panels).

- And you have already defined their Names and IDs in the PID template.

	Project	System Name	System ID	Subsystem Name	Subsystem ID	Component Type Name	Component Type ID
1							
2						Unit, Panel Assemblies PD2	
3	D	FD1-HD HVS	005	CPA	020	CPA Panel Assembly PD2 US	01000
4	D	FD1-HD HVS	005	CPA	020	Upper CPA Unit PD2 US	01100
5	D	FD1-HD HVS	005	CPA	020	Middle CPA Unit PD2 US	01200
6	D	FD1-HD HVS	005	CPA	020	Lower CPA Unit PD2 US	01300
7	D	FD1-HD HVS	005	CPA	020	CPA Panel Assembly PD2 Center	02000

**One more thing we need to go through
before diving into the WEB UI**

User Types

- **Architect(s)** : Jim Stewart, Norm Buchanan, Paul Laycock
 - **Administrators** : YOU, the liaisons from each consortia
 - **Ordinary Users** : you, your postdocs/students... etc.
 - ▶ Roles of ordinary users could be further restricted by “**Roles**”.
- Will describe what Role is later.

Flow of DB entry

1. An architect create **Component Types** in HWDB based on the **PID templates you provide.**
2. **Administrators** complete those created **Component Types.**
3. **Ordinary users** can start to enter **Items.**

Today's task!

The two Databases


Development version and Production version

- Production version : <https://dbweb0.fnal.gov/cdb/login/sso>
- Development version : <https://dbweb0.fnal.gov/cdbdev/login/sso>

You can access to both databases easily through your web browsers

Today we'll use the **development version**,
but the usage of the two databases are identical.

Opening page



DUNE Hardware DB
DEEP UNDERGROUND
NEUTRINO EXPERIMENT

Hardware DB

Table of contents

REST API

- COMPONENT TYPES
 - /component-types[?<page=<int>][&term=<pattern>]]
 - /component-types
 - /component-types/<type_name>[?history=true]
 - /component-types/<type_name>/connectors
 - /component-types/<type_name>/images
 - /component-types/<type_name>/specifications
- COMPONENTS
 - /component-types/<name>/components[?<page=<int>][&term=<pattern>]]
 - /component-types/<name>/components
 - /components/<external-id>[?history=true]
 - /components/<external-id>/container[?history=true]
 - /components/<external-id>/subcomponents[?history=true]
- TEST TYPES
 - /component-types/<type_name>/test-types
 - /component-types/<type_name>/test-types
 - /component-types/<type_name>/test-types/<test-type-name>[?history=true]
- TESTS
 - /components/<external-id>/tests[?history=true]
 - /components/<external-id>/tests
 - /components/<external-id>/tests/<test-name>[?history=true]
- OPERATIONS
 - /structures[?page=<int>]
 - /structures/<external-id>[?history=true]
 - /structures/<external-id>
- CABLES
 - /cables[?page=<int>]
 - /cables/<external-id>[?history=true]
 - /cables/<external-id>
- IMAGES
 - /images/components/<eid>
 - /images/component-types/<oid>
 - /images/component-tests/<oid>

Home

Batches

Cable Structures

Component Types

Items

Geographic Locations

Images

Manufacturers

Purchase Orders

Structures

Admin <

Logout

Overview


The components database tracks parts and what other, if any, part is attached to it. The database is designed around physicists describing what each part is and how those parts are fit together. First one must create a "type" which defines what a part is and the data collected for it. Then the data is entered for a specific "type". Tests may also be defined for a specific product "type", and data entered for each physical piece. Each physical part tracked must have some external id, which is unique across the experiment, entered with that part.

Vincit qui se vincit.

He conquers who conquers himself.

REST API


A REST API is available for retrieving and adding data. Using the API requires a certificate. It also requires you to be recognized as a member of DUNE. Be aware of the requirements for CILogon certificates.



Shows examples of REST commands

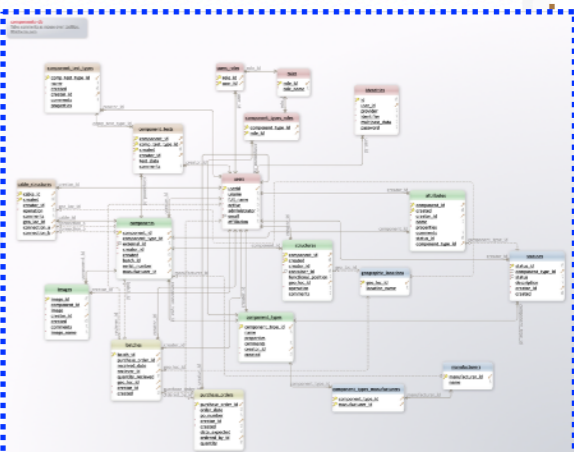
Database Schema

This product is created over a PostgreSQL database. Two views of the schema are available. One which shows all the tables. There is also a second simpler view of the schema which does not show the security tables and their connections.



Full Schema is provided in HTML

Go to Component Types



Let's EDIT a Component Type through the WEB UI

- By this time, our Architect must have created your Component Types in HWDB.
- Pick one that you like to modify.
- We'll take the Component Type, "CPA Parts FR4 bottom frame" as an example here.

Finding your Component Type

DUNE Hardware DB
DEEP UNDERGROUND NEUTRINO EXPERIMENT

Component Types

Type name	Category	Test Types	Items	Comments	Created
Z.Sandbox.VMP.Box1x1x1	generic			Yet another box to test	2021-04-07 16:13:45-05
Z.Sandbox.VMP.Box1x1x2	generic			Box 1x1x2 feet	2021-04-07 17:27:37-05
Z.Sandbox.VMP.Cable_LEMO-LEMO	cable			Testing the cables	2020-10-23 17:01:21-05
Z.Sandbox.Sandbox.CPA	generic			a placeholder for the entire CPA	2020-10-23 10:27:32-05
Z.Sandbox.Sandbox.CPA_FC	generic				2021-08-14 12:59:47-05
Z.Sandbox.Sandbox.CPA_Panel_Center_PD2	generic				2021-12-22 10:16:21-06
Z.Sandbox.Sandbox.CPA_Panel_DS_PD2	generic				2021-12-22 10:16:21-06
Z.Sandbox.Sandbox.CPA_Panels	generic				
Z.Sandbox.Sandbox.CPA_Panel_US_PD2	generic				
Z.Sandbox.Sandbox.CPA_Parts	generic			a placeholder for CPA raw parts	
Z.Sandbox.Sandbox.CPA_Parts_ARRod	generic				
Z.Sandbox.Sandbox.CPA_Parts_ARRod_Bottom	generic			Anti Rotation Rods: Bottom	
Z.Sandbox.Sandbox.CPA_Parts_ARRod_G10_Nuts	generic			Anti Rotation Rods: G10 Nuts	
Z.Sandbox.Sandbox.CPA_Parts_ARRod_Top	generic			Anti Rotation Rods: Top	
Z.Sandbox.Sandbox.CPA_Parts_BrassHdwre	generic			Brass connection parts for and FSS	
Z.Sandbox.Sandbox.CPA_Parts_BrassHdwre_Connection_Plate	generic			Brass Hardware: Connection Plate	
Z.Sandbox.Sandbox.CPA_Parts_BrassHdwre_Electrical_Strap	generic			Brass Hardware: Electrical Strap	
Z.Sandbox.Sandbox.CPA_Parts_BrassHdwre_Electrical_Tab	generic			Brass Hardware: Electrical Tab	
Z.Sandbox.Sandbox.CPA_Parts_BrassHdwre_Electrical_T_Strap	generic			Brass Hardware: Electrical Strap	
Z.Sandbox.Sandbox.CPA_Parts_FR4	generic			a placeholder for the CPA Parts FR4 frames	
Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom	generic			FR4 Frames : Bottom Support Bars	2021-11-17 11:53:35-05
Z.Sandbox.Sandbox.CPA_Parts_FR4_Intermediate	generic				2021-11-17 13:35:24-06
Z.Sandbox.Sandbox.CPA_Parts_FR4_Intermediate_A	generic			FR4 Frames : Intermediate Bars (DFD-20-A405)	2021-03-25 11:51:56-05
Z.Sandbox.Sandbox.CPA_Parts_FR4_Lower_Side_LH_A	generic			FR4 Frames : Lower Side Bars (LH) (DFD-20-A602)	2021-03-25 11:52:13-05
Z.Sandbox.Sandbox.CPA_Parts_FR4_Lower_Side_LH_B	generic			FR4 Frames : Lower Side Bars (LH) (DFD-20-B602)	2021-03-25 11:52:24-05

Apply filters

Name

Category

Comments

Creator

Part_type

- If you know the **type id** that you are looking for, enter it in “Part_type”.
(NOT for now, for most of us)
- In “Name”, you can give a part of your Component Type name.
It is Case-Sensitive.

Finding your Component Type

Apply filters

Name

Category

Comments

Creator

Part_type

DUNE Hardware DB
DEEP UNDERGROUND
NEUTRINO EXPERIMENT

Home

Batches

Cable Structures

Component Types

Items

Geographic Locations

Images

Logout

Component Types

< Previous

Next >

Type name	Category	Test Types	Items	Comments	Created	Creator	Part_type_id
Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom	generic			FR4 Frames : Bottom Support Bars	2021-03-25 11:53:35-05	Hajime Muramatsu	Z00100100017

Click the Component Type Name to go to its Type Definition page.

Completing Component Type

Edit Component Type

SPECS LOG IMAGES

Type Name CPA_Parts_FR4_bottom

Type ID 00017

Full Name Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom

Part Type ID Z00100100017

Comments FR4 Frames : Bottom Support Bars

Category generic

Managed by × type-manager

Manufacturers × Hajime Inc

Created 2021-03-25 11:53:35

Created by Hajime Muramatsu

Specifications

Version	1
Created	2021-03-25 11:53:35-05:00
Created by	Hajime Muramatsu
Datasheet	Parts ID: -1 Frame Name: Bottom Support Bar Drawing Number: DFD-20-A601 Number Ordered: -1 Number Received: -1

Connectors

SAVE DONE

- Architect must have created them for you.
 - ▶ The Type Name and ID should be identical to what you specified in your PID template.
 - ▶ Similarly for the “Full Name”, which just attaches Project/System/Subsystem names.
(This Component Type name was created temporarily)
 - ➔ Z = Project
 - ➔ Sandbox = System name
 - ➔ Sandbox = Subsystem name
 - ➔ CPA_Parts_FR4_bottom = Component Type name
 - ➔ And similarly, for Type ID.
- Notice that there is only Type ID.
No PID (or EID), yet, as this is just a Type, not an Item.

Completing Component Type

- Roles -

Edit Component Type

SPECS LOG IMAGES

Type Name CPA_Parts_FR4_bottom

Type ID 00017

Full Name Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom

Part Type ID Z00100100017

Comments FR4 Frames : Bottom Support Bars

Category generic

Managed by × type-manager

Manufacturers × Hajime Inc

Created 2021-03-25 11:53:35

Created by Hajime Muramatsu

Specifications

Version	1
Created	2021-03-25 11:53:35-05:00
Created by	Hajime Muramatsu

Datasheet

Parts ID: -1
Frame Name: Bottom Support Bar
Drawing Number: DFD-20-A601
Number Ordered: -1
Number Received: -1

Connectors

select a type

SAVE DONE

- Managed by:

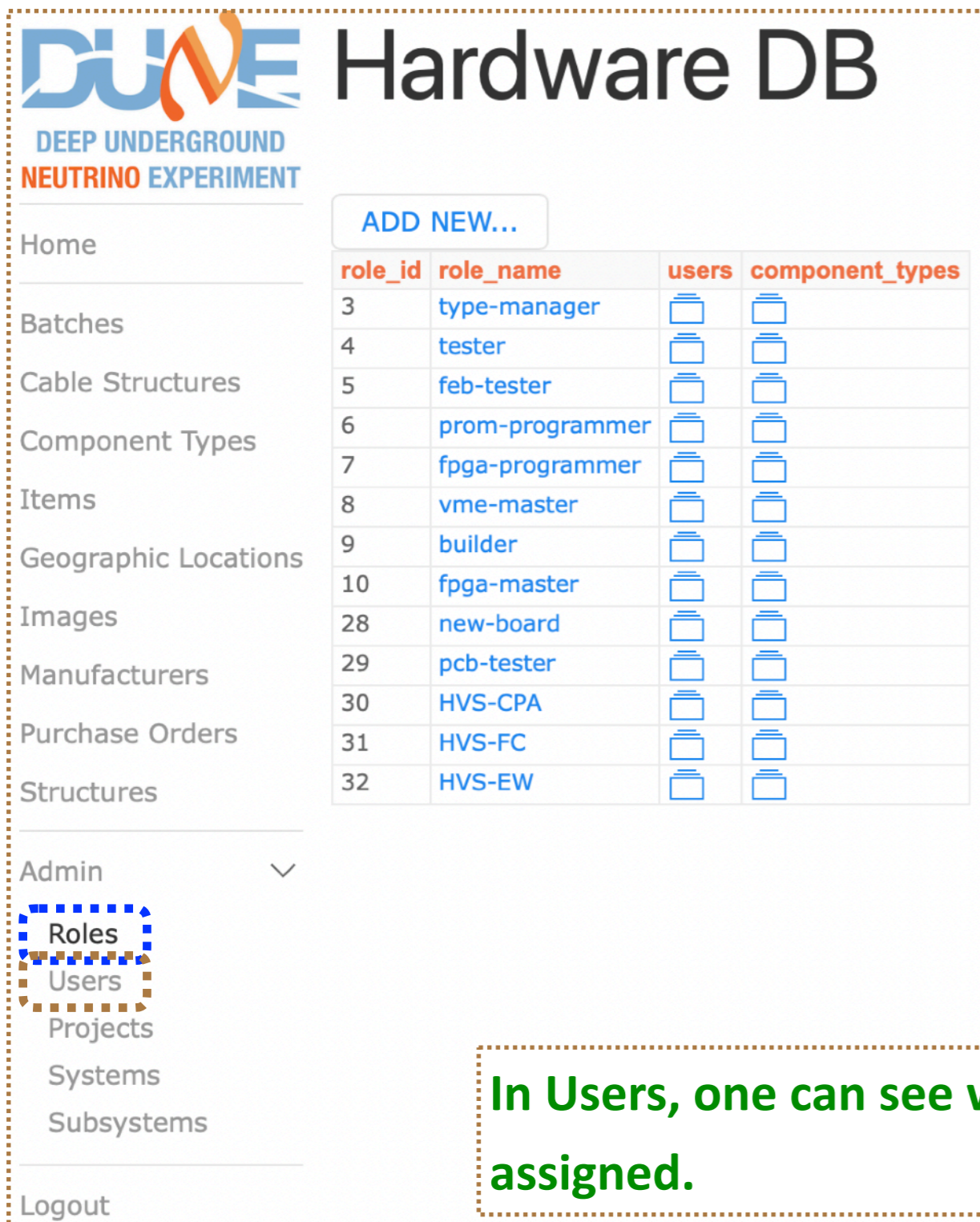
One can restrict who can edit the corresponding Items.

In this case, “type-manager” is chosen. All CPA_Parts_FR4_bottom items can be modified only by users with “type-manager” role.

- Multiple Roles can be also assigned.

Completing Component Type

- Roles -



DUNE Hardware DB

DEEP UNDERGROUND NEUTRINO EXPERIMENT

Home

Batches

Cable Structures

Component Types

Items

Geographic Locations

Images

Manufacturers

Purchase Orders

Structures

Admin

Roles

Users

Projects

Systems

Subsystems

Logout

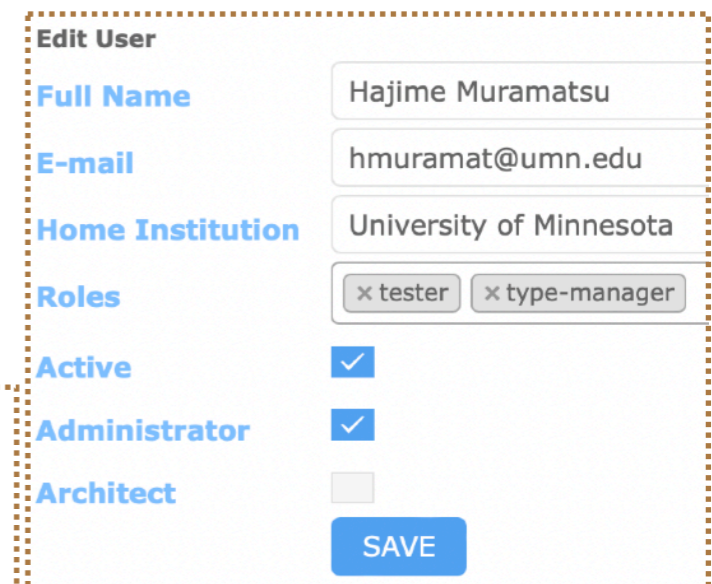
ADD NEW...

role_id	role_name	users	component_types
3	type-manager		
4	tester		
5	feb-tester		
6	prom-programmer		
7	fpga-programmer		
8	vme-master		
9	builder		
10	fpga-master		
28	new-board		
29	pcb-tester		
30	HVS-CPA		
31	HVS-FC		
32	HVS-EW		

From the side-menu, one can go to these pages and add Roles as needed.

Only Administrators are allowed to do these!!

In Users, one can see what Roles you have been assigned.



Edit User

Full Name Hajime Muramatsu

E-mail hmuramat@umn.edu

Home Institution University of Minnesota

Roles tester type-manager

Active

Administrator

Architect

SAVE

Completing Component Type

- Manufacturers -

Edit Component Type

[SPECS LOG](#) [IMAGES](#)

Type Name CPA_Parts_FR4_bottom

Type ID 00017

Full Name Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom

Part Type ID Z00100100017

Comments FR4 Frames : Bottom Support Bars

Category generic

Managed by × type-manager

Manufacturers × Hajime Inc

Created 2021-03-25 11:53:35

Created by Hajime Muramatsu

Specifications

Version	1
Created	2021-03-25 11:53:35-05:00
Created by	Hajime Muramatsu

Datasheet

Parts ID: -1
Frame Name: Bottom Support Bar
Drawing Number: DFD-20-A601
Number Ordered: -1
Number Received: -1

Connectors

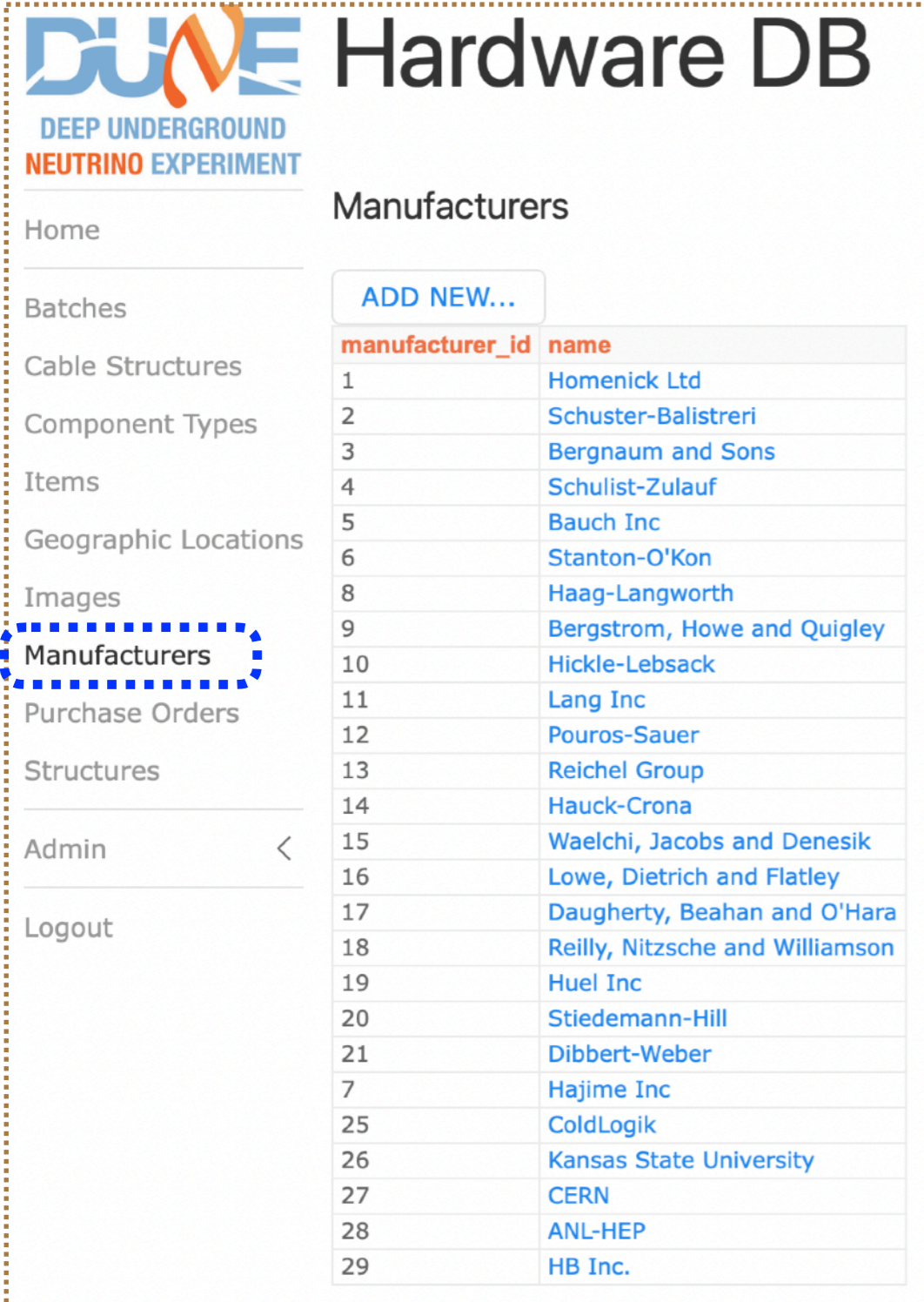
select a type

[SAVE](#) [DONE](#)

- (multiple) **Manufacturers** can be also assigned.
When a user enters an Item, s/he will be given
a list of Manufacturers that are assigned here.

Completing Component Type

- Manufacturers -



DUNE Hardware DB
DEEP UNDERGROUND
NEUTRINO EXPERIMENT

Home

Batches

Cable Structures

Component Types

Items

Geographic Locations

Images

Manufacturers

Purchase Orders

Structures

Admin <

Logout

ADD NEW...

manufacturer_id	name
1	Homenick Ltd
2	Schuster-Balistreri
3	Bergnaum and Sons
4	Schulist-Zulauf
5	Bauch Inc
6	Stanton-O'Kon
8	Haag-Langworth
9	Bergstrom, Howe and Quigley
10	Hickle-Lebsack
11	Lang Inc
12	Pouros-Sauer
13	Reichel Group
14	Hauck-Crona
15	Waelchi, Jacobs and Denesik
16	Lowe, Dietrich and Flatley
17	Daugherty, Beahan and O'Hara
18	Reilly, Nietzsche and Williamson
19	Huel Inc
20	Stiedemann-Hill
21	Dibbert-Weber
7	Hajime Inc
25	ColdLogik
26	Kansas State University
27	CERN
28	ANL-HEP
29	HB Inc.

Again, from the side-menu, one can go to these pages and add Manufacturers as needed.

Only Administrators are allowed to do these!!

Completing Component Type

- Datasheet -

Edit Component Type

SPECS LOG IMAGES

Type Name CPA_Parts_FR4_bottom

Type ID 00017

Full Name Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom

Part Type ID Z00100100017

Comments FR4 Frames : Bottom Support Bars

Category generic

Managed by × type-manager

Manufacturers × Hajime Inc

Created 2021-03-25 11:53:35

Created by Hajime Muramatsu

Specifications

Version	1
Created	2021-03-25 11:53:35-05:00
Created by	Hajime Muramatsu
Datasheet	Parts ID: -1 Frame Name: Bottom Support Bar Drawing Number: DFD-20-A601 Number Ordered: -1 Number Received: -1

Connectors

select a type

SAVE DONE

- This is the meat of Type definition.

- Provided by a markup language, called YAML.

Only a subset of YAML objects is allowed:

- ▶ Key: number or string or null
- ▶ Key: [] (an empty array)
- ▶ Key: [v0, v1, v2,..] for a dropdown selection.

See the next few pages for some examples.

Datasheet: example 1

Datasheet	Parts ID: -1
	Frame Name: Bottom Support Bar
	Drawing Number: DFD-20-A601
	Number Ordered: -1
	Number Received: -1

If you define Datasheet as “Key: value”,
like the one above,

then when you create an Item (DAY 2 material!),
you will have the corresponding box(es) to enter values.

- Box labels are the Keys you specify in your Datasheet.
- Boxes are filled with initial values, the ones you provide in your Datasheet, and they are editable.

Edit Item	
Component Type	Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom
Part ID	Z00100100017-00122
Serial Number	
Country of Origin	--SELECT--
Resp. Institution	--SELECT--
Manufacturer	--SELECT--
Batch ID	--SELECT--
Created	
Created by	
Contained in	N/A
Specifications	
Parts ID	-1
Frame Name	Bottom Support Bar
Drawing Number	DFD-20-A601
Number Ordered	-1
Number Received	-1
Sub-components	
<input type="button" value="SAVE"/> <input type="button" value="DONE"/>	

Datasheet: example 2

- One could define an array ([]). In this example, it is an empty one.
- Also, one could define Options, such as:

Gain:

- T1
- T2

(could also type as “Gain: [T1,T2]”)

This would show up with a selection box for Gain box, when you create its Item (DAY 2 material). See below:

Datasheet
AR: []
Gain:
- T1
- T2
Size: null
Type: null

Specifications

An empty array, which you can edit.

AR	[]
Gain	--SELECT--
Size	
Type	--SELECT--

AR	[]
Gain	--SELECT--
Size	
Type	--SELECT--
	T1
	T2

Datasheet: example 3

Datasheet	Std: []
	Chan: []
	Mean: []
	Nent: []
	ChipSN: []
	runtime: []
	ChanName: []
	io_group: []
	io_channel: []

- Or a list of arrays...

(Please ignore the names of Keys.

This is just an example to show
how Datasheet can be written)

Std	[]
Chan	[]
Mean	[]
Nent	[]
ChipSN	[]
runtime	[]
ChanName	[]
io_group	[]
io_channel	[]

and it appears like this,

when you create its Item (DAY 2 material).

Not sure if you want to do this manually though...

Datasheet: example 4

- Or could be a list of Keys, in which each Keys have a list of arrays...
In this example, the Key starts from ASIC_001, ASIC_002..etc.
(Please ignore the names of Keys.
This is just an example to show how Datasheet can be written)

Datasheet

```

ASIC_001:
  Std: []
  Chan: []
  Mean: []
  Nent: []
  ChipSN: 1L10451
  runtime: []
  ChanName: []
  io_group: []
  io_channel: []
ASIC_002:
  Std: []
  Chan: []
  Mean: []
  Nent: []
  ChipSN: 1L10453
  runtime: []
  ChanName: []
  io_group: []
  io_channel: []
ASIC_003:
  Std: []
  
```

And when you enter this Item, it would look like this.

ASIC_001	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10451', 'runtime': [], 'Cha
ASIC_002	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10453', 'runtime': [], 'Cha
ASIC_003	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10455', 'runtime': [], 'Cha
ASIC_004	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10456', 'runtime': [], 'Cha
ASIC_005	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10456', 'runtime': [], 'Cha
ASIC_006	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10457', 'runtime': [], 'Cha
ASIC_007	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10457', 'runtime': [], 'Cha
ASIC_008	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10458', 'runtime': [], 'Cha
ASIC_009	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10458', 'runtime': [], 'Cha
ASIC_010	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10459', 'runtime': [], 'Cha
ASIC_011	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10459', 'runtime': [], 'Cha
ASIC_012	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10460', 'runtime': [], 'Cha
ASIC_013	{'Std': [], 'Chan': [], 'Mean': [], 'Nent': [], 'ChipSN': '1L10460', 'runtime': [], 'Cha

Datasheet: example 4.1

- Similar to the previous one... **Nested Keys**

And when you enter this Item, it would look like this.

Test_A	{'Test_A_A': {'Test_A_A_1': [], 'Test_A_A_2': []}, 'Test_A_B': {'Test_A_B_1': [], 'Test_A_B_2': []}}
Test_B	{'Test_B_A': {'Test_B_A_1': [], 'Test_B_A_2': []}, 'Test_B_B': {'Test_B_B_1': [], 'Test_B_B_2': []}}

Can't see them all, unless you scroll in the boxes...

When expanded, they would look like this:

```

Test_A: {
  'Test_A_A': {
    'Test_A_A_1': [],
    'Test_A_A_2': []
  },
  'Test_A_B': {
    'Test_A_B_1': [],
    'Test_A_B_2': []
  }
}
Test_B: {
  'Test_B_A': {
    'Test_B_A_1': [],
    'Test_B_A_2': []
  },
  'Test_B_B': {
    'Test_B_B_1': [],
    'Test_B_B_2': []
  }
}

```

Test_A:
 Test_A_A:
 Test_A_A_1: []
 Test_A_A_2: []
 Test_A_B:
 Test_A_B_1: []
 Test_A_B_2: []
 Test_B:
 Test_B_A:
 Test_B_A_1: []
 Test_B_A_2: []
 Test_B_B:
 Test_B_B_1: []
 Test_B_B_2: []

Datasheet

Completing Component Type

- Connectors -

Edit Component Type

[SPECS LOG](#) [IMAGES](#)

Type Name CPA_Parts_FR4_bottom

Type ID 00017

Full Name Z.Sandbox.Sandbox.CPA_Parts_FR4_bottom

Part Type ID Z00100100017

Comments FR4 Frames : Bottom Support Bars

Category generic

Managed by × type-manager

Manufacturers × Hajime Inc

Created 2021-03-25 11:53:35

Created by Hajime Muramatsu

Specifications

Version	1
Created	2021-03-25 11:53:35-05:00
Created by	Hajime Muramatsu
Datasheet	Parts ID: -1 Frame Name: Bottom Support Bar Drawing Number: DFD-20-A601 Number Ordered: -1 Number Received: -1

Connectors

[SAVE](#) [DONE](#)

One can connect “sub-Component Type” to this Component Type.

... what is a sub-Component!?

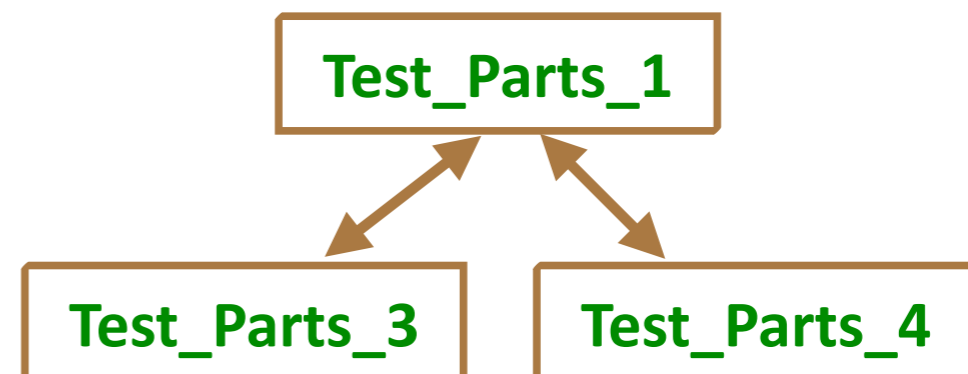
Completing Component Type

- Connectors -

- This is the **Relation** we referred to earlier in this session. Again, for instance, an assembly is a **Component Type** that is constructed from other **sub-Component Types** (E.g., A Component Type, CPA Plane, is made of sub-Component Types, CPA Panels).

- In the following example, we employ 3 Component Types:

- ▶ Test_Parts_1
- ▶ Test_Parts_3
- ▶ Test_Parts_4



- And we want;

- ▶ Test_Parts_3 and Test_Parts_4 to be **sub-components** of Test_Parts_1

Defining sub-Component Types

WANT Test_Parts_3 and Test_Parts_4 to be sub-components of Test_Parts_1

Edit Component Type

SPECS LOG IMAGES

Type Name Test_Parts_1

Type ID 00048

Full Name Z.Sandbox.Sandbox.Test_Parts_1

Part Type ID Z00100100048

Comments

Category generic

Managed by x type-manager

Manufacturers x CERN x Hajime Inc

Created 2022-04-21 17:29:00

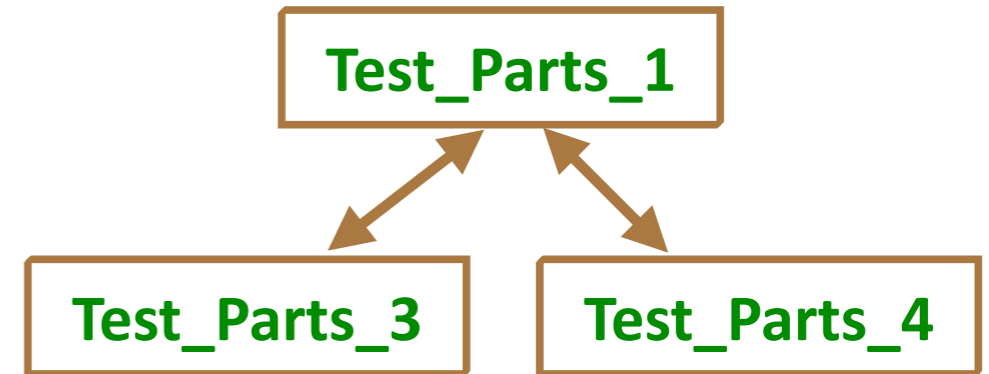
Created by Hajime Muramatsu

Specifications

Version	1
Created	2022-02-08 08:19:19-06:00
Created by	Hajime Muramatsu
Datasheet	ChipSN: '-----'

Connectors select a type + -

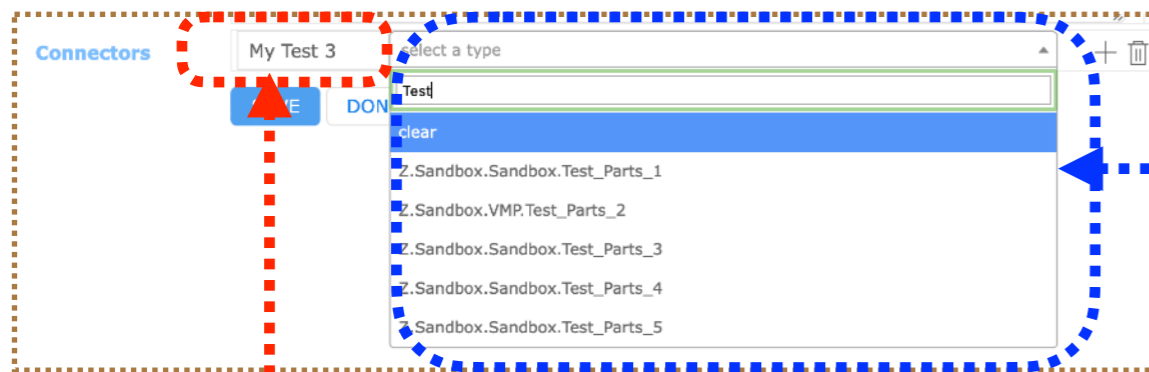
SAVE DONE



- We'll add sub-Component Types to the Component Type, Test_Parts_1.

Defining sub-Component Types

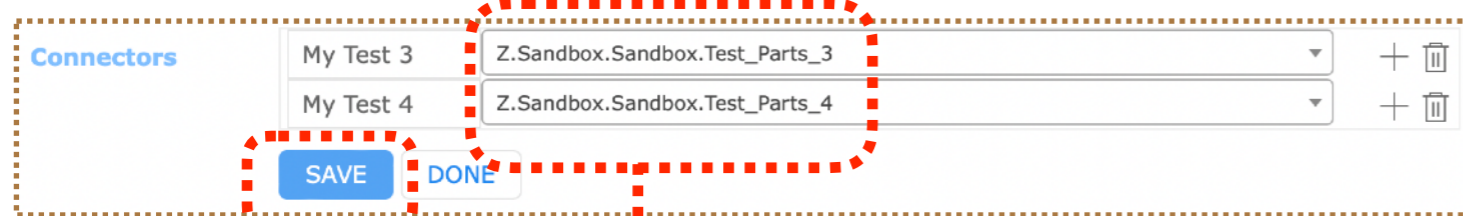
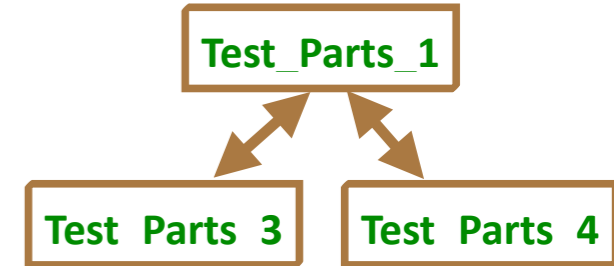
WANT Test_Parts_3 and Test_Parts_4 to be sub-components of Test_Parts_1



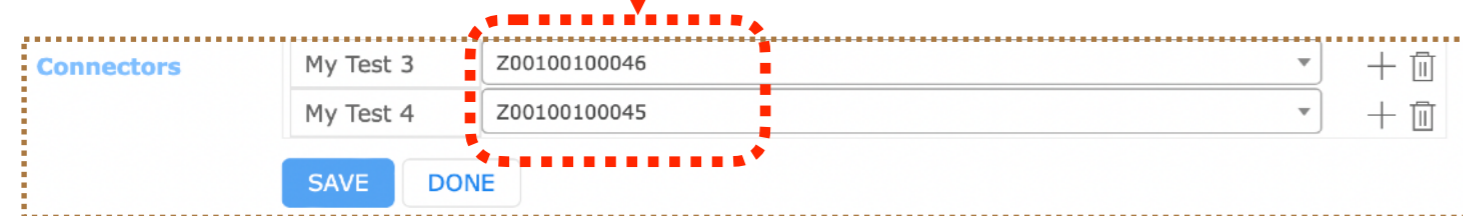
Provide the corresponding sub-Component Type Name.

Entering 2 characters will give you a list of matched candidates.

Give your preferred name for its functional position (can be any, but needs to be unique within this Component Type)



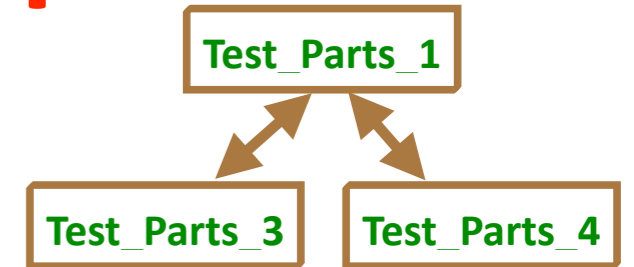
Now the two Type Names are entered.



Saving triggers the conversion from Type Names to Type IDs.

That is it! You are done!

Completing Component Type - Connectors -



Now when you enter an Item for this type (Test_Parts_1) you can add actual existing sub-Components like below (DAY 2 material).

Edit Item Z00100100048-00038

QR SPECS LOG STRUCTURE LOG CONTAINER LOG TEST LOG IMAGES

Component Type Z.Sandbox.Sandbox.Test_Parts_1

Part ID Z00100100048-00038-US186

Serial Number

Country of Origin United States

Resp. Institution University of Minnesota Twin Cities

Manufacturer Hajime Inc

Batch ID --SELECT--

Created 2022-04-28 17:44:01

Created by Hajime Muramatsu

Contained in N/A

Specifications

ChipSN testing - 2

Sub-components

My Test 3:Test_Parts_3	Z00100100046-00001
My Test 4:Test_Parts_4	Z00100100045-00004

SAVE DONE

Edit Item Z00100100045-00004

QR SPECS LOG STRUCTURE LOG CONTAINER LOG TEST LOG IMAGES

Component Type Z.Sandbox.Sandbox.Test_Parts_4

Part ID Z00100100045-00004-US001

Serial Number

Country of Origin United States

Resp. Institution Fermi National Accelerator Laboratory

Manufacturer Hajime Inc

Batch ID 1237

Created 2021-11-01 08:42:18

Created by Hajime Muramatsu

Contained in Z00100100048-00038

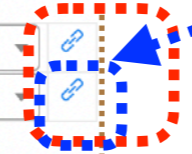
Specifications

Sub-components

SAVE DONE

Parent Item

A daughter Item



Links to the corresponding Items will also show up.

History of your Specifications

Now your Component Type page should look like this.

One can also see a modification history to your Specifications.

Edit Component Type

SPECS LOG IMAGES

Type Name Test_Parts_1

Type ID 00048

Full Name Z.Sandbox.Sandbox.Test_Parts_1

Part Type ID Z00100100048

Comments Testing...

Category generic

Managed by x tester x type-manager

Manufacturers x CERN x Hajime Inc

Created 2022-04-25 19:26:37

Created by Hajime Muramatsu

Specifications

Version	Created	Created by	Datasheet
7	2022-04-25 19:30:06-05:00	Hajime Muramatsu	ChipSN: testing Type...

Connectors

My Test 3	Z00100100046	+ -
My Test 4	Z00100100045	+ -

SAVE DONE

The latest on the top, with a version #.

Specification history for Test_Parts_1

Created	Creator	Version	Datasheet
2022-04-25 19:30:06-05:00	Hajime Muramatsu	7	ChipSN: testing Type...
2022-04-25 19:27:38-05:00	Hajime Muramatsu	6	ChipSN: testing Type...2
2022-04-25 19:26:37-05:00	Hajime Muramatsu	5	ChipSN: testing Type...
2022-04-25 19:24:56-05:00	Hajime Muramatsu	4	ChipSN: testing Type
2022-04-25 08:58:47-05:00	Hajime Muramatsu	3	ChipSN: testing Type...
2022-04-22 13:40:50-05:00	Hajime Muramatsu	2	ChipSN: testing Type...
2022-02-08 08:19:19-06:00	Hajime Muramatsu	1	ChipSN: '------'
2021-10-15 08:22:45-05:00	Hajime Muramatsu	0	

REST API

- In the following examples, we'll use a **curl** command.
- **curl** is a command-line tool to transfer data to or from a server.
And it supports the protocol we need, **https**.
- A typical usage is like;
curl [options] [URL...]

REST API

- In the rest of this talk, I'll show a bunch of command lines, which usually starts with the following.

```
curl --cert-type P12 --cert MyCert.p12:myPSWD 'https://dbwebapi2.fnal.gov:8443/cdbdev/api/...'
```

Here,

- ▶ MyCert.p12 is a p12 certificate, obtained from <https://cilogon.org>
- ▶ myPSWD is the “its password”.

- Since they'll show up repeatedly, I will abbreviate them in the following way:

- ▶ `curl --cert-type P12 --cert MyCert.p12:myPSWD` → **CURL**
- ▶ `https://dbwebapi2.fnal.gov:8443/cdbdev/api` → **APIPATH**

Updating a Component Type (PATCH)

Edit Component Type

SPECS LOG IMAGES

Type Name Test_Parts_1

Type ID 00048

Full Name Z.Sandbox.Sandbox.Test_Parts_1

Part Type ID Z00100100048

Comments Testing...

Category generic

Managed by × tester × type-manager

Manufacturers × CERN × Hajime Inc

Created 2022-04-21 17:44:56

Created by Hajime Muramatsu

Specifications

Version	3
Created	2022-04-25 08:58:47-05:00
Created by	Hajime Muramatsu

Datasheet

ChipSN: testing Type...

Connectors

MyTest	Z00100100046	+ -
--------	--------------	-----

SAVE DONE

- Again, the Architect must have created your Component Types.
In this example, its Type Name is “Test_Parts_1”.
- Want to define:
 - ▶ “Managed by”
 - ▶ “Manufacturers”
 - ▶ “Datasheet”
 - ▶ “Connectors”

Let's do these with the API!

Updating a Component Type (PATCH)

- The API endpoint : `/api/component-types/<type_id>`

- An example of actual line:

```
CURL -H "Content-Type: application/json" -X PATCH -d @Patch_CompType_Test_parts_1.json  
'APIPATH/component-types/Z00100100048'
```

- need to PATCH (not POST)

- In this example, the Component Type ID is Z00100100048

- The JSON file looks like this:

Notice that you can set up **connectors**

(**sub-Component Type**) here as well,

by specifying its **Component Type ID(s)**.

- When executed, you should see a response like this;

```
{  
  "data" : "Updated",  
  "id" : 152,  
  "part_type_id" : "Z00100100048",  
  "status" : "OK"  
}
```

```
{  
  "part_type_id": "Z00100100048",  
  "comments": "Testing...",  
  "manufacturers": [7,27],  
  "roles": [3,4],  
  "properties": {  
    "specifications": {  
      "ChipSN": "testing Type..."  
    }  
  },  
  "connectors": {  
    "MyTest": "Z00100100046"  
  }  
}
```

by the way...

- To format responses from API human-readable, I use `json_opt` command, like this:

```
CURL -H "Content-Type: application/json" -X PATCH -d @Patch_CompType_Test_parts_1.json  
'APIPATH/component-types/Z00100100048/components' | json_pp -json_opt pretty,canonical
```

Checking a Component Type (GET)

- The API endpoint : `/api/component-types/<type_id>`
- An example of actual line:
`CURL 'APIPATH/component-types/Z00100100048'`

The response is long.. see the next page

```
{
  "data" : {
    "category" : "generic",
    "comments" : "Testing...",
    "connectors" : {
      "MyTest" : "Z00100100046"
    },
    "created" : "2022-04-21T17:44:56.091295-05:00",
    "creator" : {
      "id" : 12624,
      "name" : "Hajime Muramatsu"
    },
    "full_name" : "Z.Sandbox.Sandbox.Test_Parts_1",
    "id" : 152,
    "manufacturers" : [
      {
        "id" : 7,
        "name" : "Hajime Inc"
      },
      {
        "id" : 27,
        "name" : "CERN"
      }
    ],
    "properties" : {
      "specifications" : [
        {
          "created" : "2022-04-25 08:58:47-05:00",
          "creator" : "Hajime Muramatsu",
          "datasheet" : {
            "ChipSN" : "testing Type..."
          },
          "version" : 3
        }
      ]
    },
    "roles" : [
      {
        "id" : 4,
        "name" : "tester"
      },
      {
        "id" : 3,
        "name" : "type-manager"
      }
    ],
    "subsystem" : {
      "id" : 8,
      "name" : "Sandbox"
    }
  },
}
```

Here is the “data” blob. It’s there..

We see;

- comments
- manufacturers
- roles (=“Managed by”)
- specifications
- subsystem
- connectors (sub-component Type ID)

It shows the latest version of Specifications.

If there are multiple versions, you could do

CURL 'APIPATH/component-types/Z00100100048?history=true'

This will show the all versions of Specifications.

There are other blobs as you can see.

See the next page.

```
"link" : {
  "href" : "/cdbdev/api/component-types/Z00100100048",
  "rel" : "self"
},
"methods" : [
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/components",
    "rel" : "Components"
  },
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/test-types",
    "rel" : "Test Types"
  },
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/connectors",
    "rel" : "Connectors description"
  },
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/specifications",
    "rel" : "Type specs"
  },
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/images",
    "rel" : "Images"
  },
  {
    "href" : "/cdbdev/api/component-types/Z00100100048/bulk-add",
    "rel" : "Bulk Add"
  }
],
"status" : "OK"
```

**There are other blobs as you can see:
link, methods, and status**

link:

- shows the href you just accessed

methods:

- other API available that is associated with this Component Type.

We'll get to some of them on DAY 2.

status:

- reports a status of the access you just made.

List of API endpoints we covered today

- **PATCH a Component Type** : `/api/component-types/<type_id>`
- **GET a Component Type** : `/api/component-types/<type_id>`

List of API endpoints we will cover on DAY 2

- **POST/GET an Item(s) : /api/component-types/<type_id>/components**
- **GET an Item : /api/components/<eid>**
- **POST Items : /api/component-types/<type_id>/bulk-add**
- **PATCH/GET a sub-component(s) : /api/components/<eid>/subcomponents**
- **POST/GET a Test Type(s) : /api/component-types/<type_id>/test-types**
- **POST a Test : /api/components/<eid>/tests**
- **GET a Test : /api/components/<eid>/tests/<test_type_name>**
- **GET a bar-code : /api/get-barcode/<cid>**
- **GET a QR-code : /api/get-qrcode/<cid>**
- **POST/GET an Image/info : /api/components/<eid>/images**
- **GET an Image : /api/img/<image_id>**

Try to complete other Component Type definitions

- Do these with **both** WEB UI and REST API.

Also;

- add new Roles and assign them to you (and your co-workers)
- add new Manufacturers and assign them to your Component Types.

And please do these.

We will use YOUR assigned Roles and Manufacturers on DAY 2.

All available commands are described here:

https://cdcvs.fnal.gov/redmine/projects/components-db/wiki/Rest_API